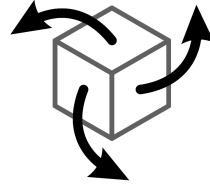


3-SPACE
SENSOR

3-SPACE



3-Space Sensor Wireless 2.4GHz

Miniature Wireless Attitude & Heading
Reference System

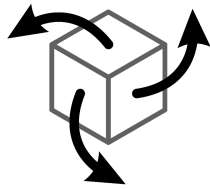
User's Manual

YEI Technology
630 Second Street
Portsmouth, Ohio 45662

www.YeiTechnology.com
www.3SpaceSensor.com

This page intentionally left blank

This page intentionally left blank



3-Space Sensor Wireless 2.4GHz

Miniature Wireless Attitude & Heading
Reference System

User's Manual

YEI Technology
630 Second Street
Portsmouth, Ohio 45662

www.YeiTechnology.com
www.3SpaceSensor.com

Toll-Free: 888-395-9029
Phone: 740-355-9029

Table of Contents

1. Usage/Safety Considerations.....	1
1.1 Usage Conditions.....	1
1.2 Technical Support and Repairs.....	1
1.3 Regulatory Approval.....	1
1.3.1 United States FCC Approval.....	1
1.3.2 Canada IC Approval.....	2
1.3.3 European Approval.....	2
1.4 Battery Safety Considerations.....	2
2. Overview of the YEI Wireless 3-Space Sensor.....	3
2.1 Introduction.....	3
2.2 Applications.....	3
2.3 Hardware Overview.....	4
2.3.1 Wireless Sensor Hardware Overview.....	4
2.3.2 Wireless Dongle Hardware Overview.....	4
2.4 Features.....	5
2.5 Block Diagram of Sensor Operation.....	6
2.6 Specifications.....	7
2.7 Physical Dimensions.....	8
2.8 Axis Assignment.....	9
2.9 Wireless Terminology.....	9
2.10 Wireless LED Modes.....	10
3. Description of the 3-Space Sensor.....	11
3.1 Orientation Estimation.....	11
3.1.1 Component Sensors.....	11
3.1.2 Scale, Bias, and Cross-Axis Effect.....	11
3.1.3 Reference Vectors.....	11
3.1.4 Kalman Filter.....	12
3.1.5 Reference Orientation/Taring.....	12
3.1.6 Other Estimation Parameters.....	12
3.2 Communication.....	13
3.3 Input Device Emulation.....	13
3.3.1 Axes and Buttons.....	13
3.3.2 Joystick.....	13
3.3.3 Mouse.....	13
3.3.4 Wireless Joystick/Mouse.....	13
3.4 Sensor Settings.....	14
3.4.1 Committing Settings.....	14
3.4.2 Committing Wireless Settings.....	14
3.4.3 Natural Axes.....	14
3.4.4 Sensor General Settings.....	14
3.4.5 Dongle General Settings.....	15
3.4.6 Sensor Wireless Settings.....	15
3.4.7 Dongle Wireless Settings.....	15
4. 3-Space Sensor Usage/Protocol.....	16
4.1 Usage Overview.....	16
4.1.1 Protocol Overview.....	16
4.1.2 Computer Interfacing Overview(USB).....	16
4.1.3 Computer Interfacing Overview(Wireless).....	16
4.2 Wired Protocol Packet Format.....	17
4.2.1 Binary Packet Format.....	17
4.2.2 ASCII Text Packet Format.....	18
4.3 Wireless Protocol Packet Format.....	19
4.3.1 Wireless Communication Format.....	19
4.3.2 Binary Packet Format.....	19
4.3.3 Binary Command Response.....	20
4.3.4 Sample Binary Commands.....	20
4.3.5 ASCII Text Packet Format.....	21
4.3.6 ASCII Command Response.....	22
4.3.7 Sample ASCII Commands.....	22
4.4 Wireless Asynchronous Protocol.....	23
4.4.1 Asynchronous Communication Format.....	23
4.4.2 Interval and Duration.....	23
4.4.3 Starting/Stopping Asynchronous Transmissions.....	24
4.4.4 Asynchronous Auto Flush.....	24
4.4.5 Asynchronous Manual Flush.....	24

4.4.6 Asynchronous Timestamps and Flush Bits.....	25
4.4.7 Sample Asynchronous Requests.....	25
4.5 Wireless Broadcast Protocol.....	26
4.5.1 Broadcast Communication Format.....	26
4.5.2 Sample Broadcasts.....	26
4.6 Command Overview.....	27
4.6.1 Commands for Reading Filtered Sensor Data.....	27
4.6.2 Commands for Reading Normalized Sensor Data.....	27
4.6.3 Commands for Reading Raw Sensor Data.....	28
4.6.4 Commands for Setting Filter Parameters.....	28
4.6.5 Commands for Reading Filter Parameters.....	29
4.6.6 Commands for Calibration.....	30
4.6.7 Commands for Wireless Sensor Units.....	30
4.3.8 Commands for Wireless Dongle Units.....	30
4.3.9 General Commands.....	31
4.6.10 HID Commands.....	31
4.4. Command Details.....	32
Appendix.....	53
USB Connector.....	53
Hex / Decimal Conversion Chart.....	53

This page intentionally left blank

This page intentionally left blank

1. Usage/Safety Considerations

1.1 Usage Conditions

- Do not use the 3-Space Sensor in any system on which people's lives depend (life support, weapons, etc.)
- Because of its reliance on a compass, the 3-Space Sensor will not work properly near the earth's north or south pole.
- Because of its reliance on a compass and accelerometer, the 3-Space Sensor will not work properly in outer space or on planets with no magnetic field.
- Care should be taken when using the 3-Space Sensor in a car or other moving vehicle, as the disturbances caused by the vehicle's acceleration may cause the sensor to give inaccurate readings.
- Because of its reliance on a compass, care should be taken when using the 3-Space Sensor near ferrous metal structures, magnetic fields, current carrying conductors, and should be kept about 6 inches away from any computer screens or towers.
- Since the Wireless 3-Space Sensor uses RF communication technology, communication failure modes should be carefully considered when designing a system that uses the wireless 3-Space Sensor.
- The Wireless 3-Space Sensor is powered by a rechargeable lithium-polymer battery. Lithium-polymer batteries have high energy densities and can be dangerous if not used properly. See section 1.4 Battery Considerations for further information pertaining to battery safety.

1.2 Technical Support and Repairs

Limited Product Warranty: YEI warrants the media and hardware on which products are furnished to be free from defects in materials and workmanship under normal use for sixty (60) days from the date of delivery. No warranties exist for any misuse. YEI will repair or replace any defective product which is returned within this time period.

Product Support: YEI provides technical and user support via our toll-free number (888-395-9029) and via email (support@YostEngineering.com). Support is provided for the lifetime of the equipment. Requests for repairs should be made through the Support department. For damage occurring outside of the warranty period or provisions, customers will be provided with cost estimates prior to repairs being performed.

1.3 Regulatory Approval

1.3.1 United States FCC Approval

This device contains FCC ID: OA3MRF24J40MA

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy, and if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

To satisfy FCC RF Exposure requirements for mobile and base station transmission devices, a separation distance of 20 cm or more should be maintained between the antenna of this device and persons during operation. To ensure compliance, operation at closer than this distance is not recommended. The antenna(s) used for this transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

If the Wireless Unit is used in a portable application (antenna is less than 20 cm from persons during operation), the integrator is responsible for performing Specific Absorption Rate (SAR) testing in accordance with FCC rules 2.1091

1.3.2 Canada IC Approval

This device contains IC ID: 7693A-24J40MA

This device has been certified for use in Canada under Industry Canada (IC) Radio Standards Specification (RSS) RSS-210 and RSS-Gen.

1.3.3 European Approval

The device contains a communication module that has been certified for use in European countries.

The following testing has been completed:

Test standard ETSI EN 300 328 V1.7.1 (2006-10):

- Maximum Transmit Power
- Maximum EIRP Spectral Density
- Frequency Range
- Radiated Emissions

Test standards ETSI EN 301 489-1:2008 and ETSI EN 301 489-17:2008:

- Radiated Emissions
- Electro-Static Discharge
- Radiated RF Susceptibility

1.4 Battery Safety Considerations

The Wireless 3-Space Sensor contains a rechargeable lithium-polymer battery. Lithium-polymer batteries have high energy densities and can be dangerous if not used and cared for properly. The Wireless 3-space Sensor has been designed to include multiple levels of battery safety assurance. The Wireless 3-Space Sensor circuitry includes smart charging circuitry with thermal management to prevent over-charging the battery. The battery pack itself also includes protection circuitry to prevent over-charge, over-voltage, over-current, and over-discharge conditions.

Most battery issues arise from improper handling of batteries, and particularly from the continued use of damaged batteries.

As with any lithium-polymer battery-powered device, the following should be observed:

- Don't disassemble, crush, puncture, shred, or otherwise attempt to change the form of your battery.
- Don't attempt to change or modify the battery yourself. Contact YEI technical support for battery replacement or battery repair.
- Don't let the mobile device or battery come in contact with water.
- Don't allow the battery to touch metal objects.
- Don't place the sensor unit near a heat source. Excessive heat can damage the sensor unit or the battery. High temperatures can cause the battery to swell, leak, or malfunction.
- Don't dry a wet or damp sensor unit with an appliance or heat source, such as a hair dryer or microwave oven.
- Don't drop the sensor unit. Dropping, especially on a hard surface, can potentially cause damage to the sensor unit or the battery.
- Discontinue use immediately and contact YEI technical support if the battery or sensor unit produce odors, emit smoke, exhibit swelling, produce excess heat, exhibit leaking.
- Dispose of Lithium-polymer batteries properly in accordance with local, state, and federal guidelines.

2. Overview of the YEI Wireless 3-Space Sensor

2.1 Introduction

The YEI 3-Space Sensor™ Wireless integrates a miniature, high-precision, high-reliability, Attitude and Heading Reference System (AHRS) with a 2.4GHz DSSS communication interface and a rechargeable lithium-polymer battery solution into a single low-cost end-use-ready unit. The Attitude and Heading Reference System (AHRS) uses triaxial gyroscope, accelerometer, and compass sensors in conjunction with advanced on-board filtering and processing algorithms to determine orientation relative to an absolute reference orientation in real-time.

Orientation can be returned in absolute terms or relative to a designated reference orientation. The proprietary multi-reference vector mode increases accuracy and greatly reduces and compensates for sensor error. The YEI 3-Space Sensor Wireless system also utilizes a dynamic sensor confidence algorithm that ensures optimal accuracy and precision across a wide range of operating conditions.

The YEI 3-Space Sensor Wireless unit features are accessible via a well-documented open communication protocol that allows access to all available sensor data and configuration parameters using either 2.4GHz DSSS wireless or USB 2.0 interfaces. Versatile commands allow access to raw sensor data, normalized sensor data, and filtered absolute and relative orientation outputs in multiple formats including: quaternion, Euler angles (pitch/roll/yaw), rotation matrix, axis angle, two vector(forward/up).

The YEI Wireless 3-Space Sensor™ communicates with a host PC via a USB dongle installed in the PC. Up to 15 sensor units can be associated with each wireless dongle, and multiple dongles can be used simultaneously to achieve high sensor counts or increase individual sensor throughput. Sensor and dongle units have individual wireless network PAN Id assignment and wireless channel assignment to allow multiple sensors to communicate simultaneously without interference or performance degradation.

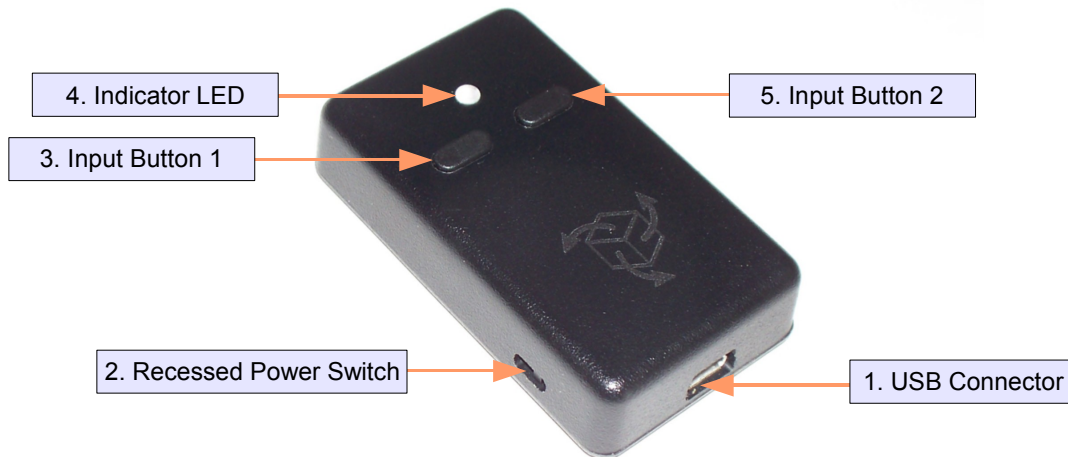
When used as a USB device, the 3-Space Sensor™ provides mouse emulation and joystick emulation modes that ease integration with existing applications.

2.2 Applications

- Robotics
- Motion capture
- Positioning and stabilization
- Vibration analysis
- Inertial augmented localization
- Personnel / pedestrian navigation and tracking
- Unmanned air/land/water vehicle navigation
- Education and performing arts
- Healthcare monitoring
- Gaming and motion control
- Accessibility interfaces
- Virtual reality and immersive simulation

2.3 Hardware Overview

2.3.1 Wireless Sensor Hardware Overview



1. **USB Connector** – The 3-Space Sensor uses a 5-pin mini USB connector to connect to a computer via USB and to charge the internal battery. The USB connector provides for both power and communication signals.
2. **Recessed Power Switch** – The 3-Space Sensor can be switched on and off when powered from the internal battery by using the recessed power switch. When connected via USB, the unit is powered and the batteries will begin recharging regardless of the position of the recessed power switch.
3. **Input Button 1** – The 3-Space Sensor includes two input buttons that can be used in conjunction with the orientation sensing capabilities of the device. The inputs are especially useful when using the 3-Space Sensor as an input device such as in joystick emulation mode or mouse emulation mode.
4. **Indicator LED** – The 3-Space Sensor includes an RGB LED that can be used for visual status feedback.
5. **Input Button 2** – The 3-Space Sensor includes two input buttons that can be used in conjunction with the orientation sensing capabilities of the device. The inputs are especially useful when using the 3-Space Sensor as an input device such as in joystick emulation mode or mouse emulation mode.

2.3.2 Wireless Dongle Hardware Overview



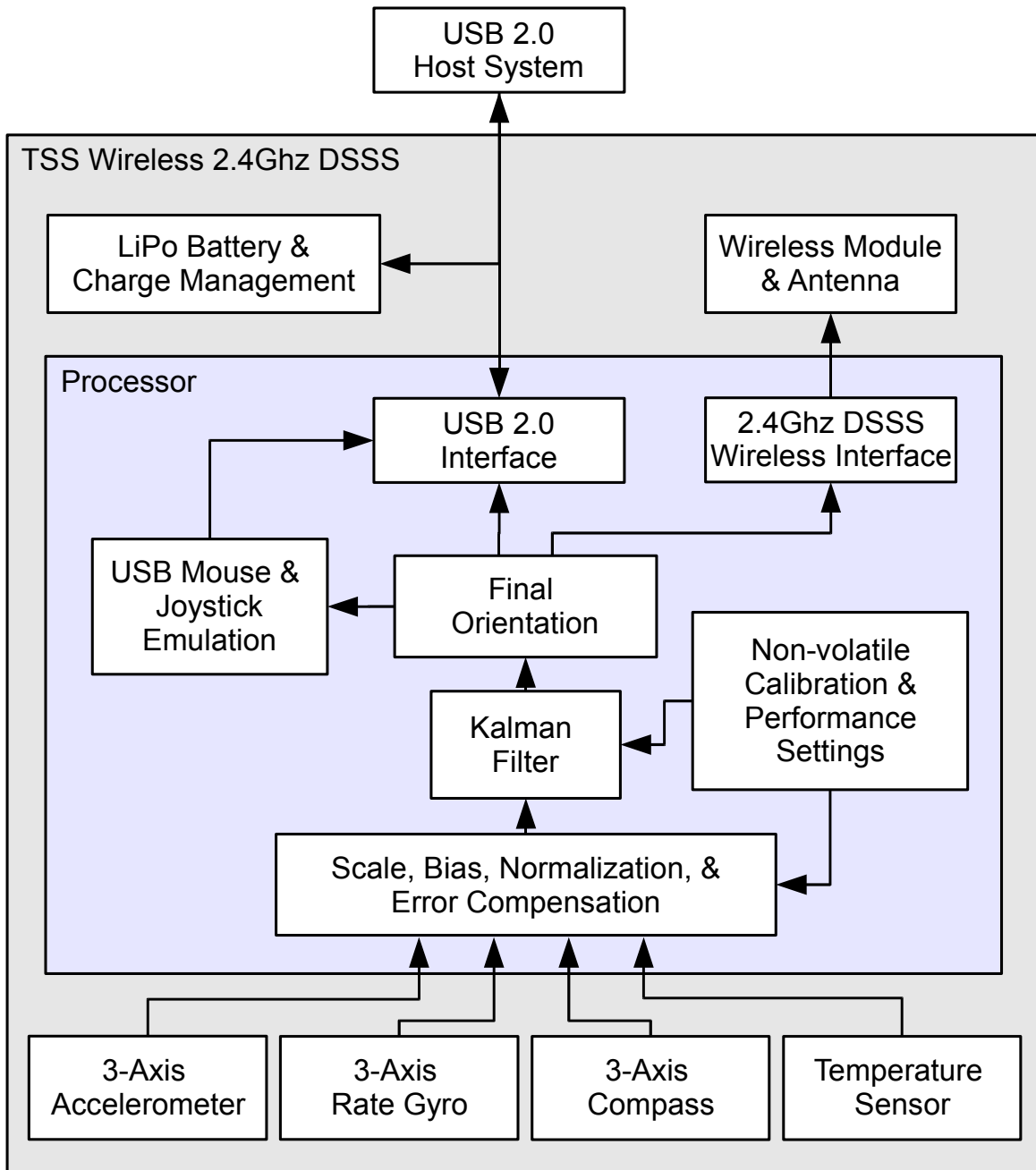
1. **USB Connector** – The 3-Space Wireless Dongle uses a 5-pin mini USB connector to connect to a computer via USB. The USB connector provides for both power and communication.
2. **Indicator LED** – The 3-Space Wireless Dongle includes an RGB LED that can be used for visual status feedback.

2.4 Features

The YEI 3-Space Sensor Wireless has many features that allow it to be a flexible all-in-one solution for your orientation sensing needs. Below are some of the key features:

- Small self-contained high-performance wireless AHRS at 35mm x 60mm x 15mm and 28 grams
- Integrated 2.4GHz DSSS wireless communication interface allows high-performance at ranges up to 200'
- Integrated Rechargeable Lithium-Polymer battery and charge control allows battery life of 5+ hours at full performance
- Fast sensor update and filter rate allow use in real-time applications, including stabilization, virtual reality, real-time immersive simulation, and robotics
- Highly customizable orientation sensing with options such as tunable filtering, oversampling, and orientation error correction
- Advanced integrated Kalman filtering allows sensor to automatically reduce the effects of sensor noise and sensor error
- Robust open protocol allows commands to be sent in human readable form, or more quickly in machine readable form
- Orientation output format available in absolute or relative terms in multiple formats (quaternion, rotation matrix, axis angle, two-vector)
- Absolute or custom reference axes
- Access to raw sensor data
- Flexible communication options: USB 2.0 or wireless 2.4GHz DSSS (FCC Certified)
- 2.4Ghz DSSS wireless communication allows orientation sensing without any wires, making activities requiring a high level of mobility like motion capture possible.
- Wireless sensors have configurable wireless channel selection and network PAN Ids to allow multiple sensors to communicate simultaneously without interference or performance degradation
- Each communication dongle unit supports up to 15 independent sensor units
- Asynchronous communication support for improved performance with multiple sensor units
- Communication through a virtual COM port
- USB joystick/mouse emulation modes ease integration with existing applications
- Upgradeable firmware
- RGB status LED, two programmable input buttons
- Available in either hand-held or strap-down packaging
- RoHS compliant

2.5 Block Diagram of Sensor Operation



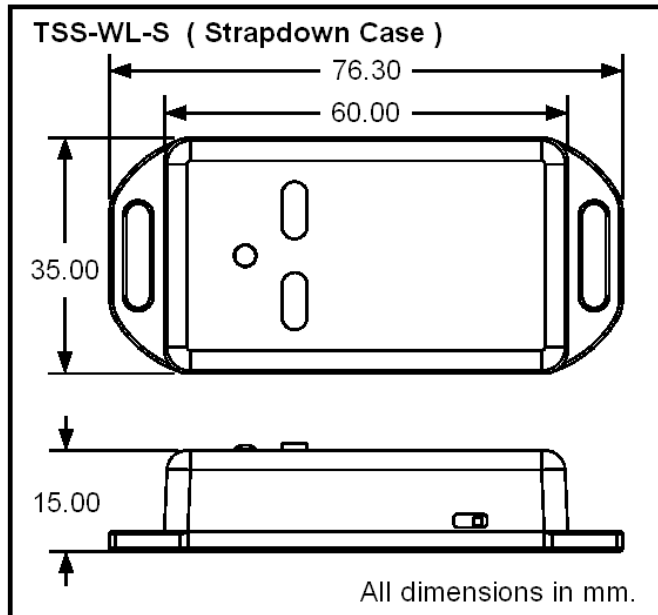
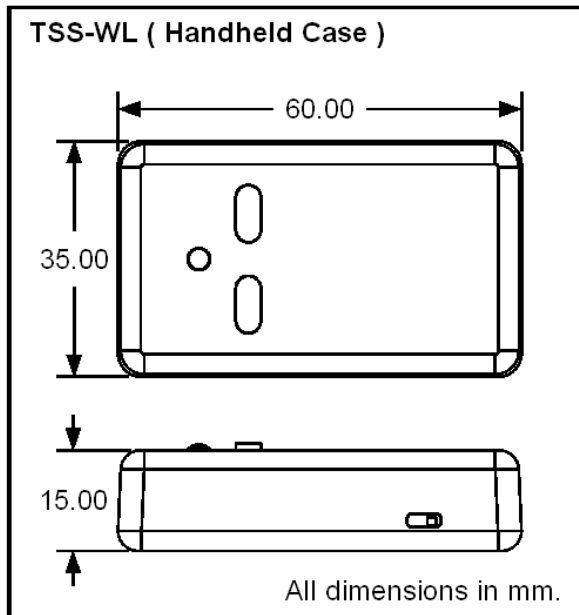
2.6 Specifications

General	
Part number	TSS-WL (Handheld Sensor Unit) TSS-WL-S (Strapdown Sensor Unit)
Dimensions	35mm x 60mm x 15mm (1.38 x 2.36 x 0.59 in.)
Weight	28 grams (0.98 oz)
Supply voltage	+5v USB
Battery technology	rechargeable Lithium-Polymer
Battery lifetime	5+ hours continuous use at full performance
Communication interfaces	USB 2.0, 2.4GHz DSSS Wireless (FCC certified)
Wireless communication range	up to 200'
Wireless PAN Ids selectable	65536
Wireless channels selectable	16 (2.4GHz channel 11 through 26)
Filter update rate	up to 200Hz with full functionality
Orientation output	absolute & relative quaternion, Euler angles, axis angle, rotation matrix, two vector
Other output	raw sensor data, normalized sensor data, temperature
Shock survivability	5000g
Temperature range	-40C ~ 85C (-40F ~ 185F)
Processor	32-bit RISC running @ 60MHz
Sensor	
Orientation range	360° about all axes
Orientation accuracy	±2° for dynamic conditions & all orientations
Orientation resolution	<0.08°
Orientation repeatability	0.085° for all orientations
Accelerometer scale	±2g / ±4g / ±8g selectable
Accelerometer resolution	14 bit
Accelerometer noise density	99µg/√Hz
Accelerometer sensitivity	0.00024g/digit for ±2g range 0.00048g/digit for ±4g range 0.00096g/digit for ±8g range
Accelerometer temperature sensitivity	±0.008%/°C
Gyro scale	±250/±500/±2000 °/sec selectable
Gyro resolution	16 bit
Gyro noise density	0.03°/sec/√Hz
Gyro bias stability @ 25°C	11°/hr average for all axes
Gyro sensitivity	0.00875°/sec/digit for ±250°/sec 0.01750°/sec/digit for ±500°/sec 0.070°/sec/digit for ±2000°/sec
Gyro non-linearity	0.2% full-scale
Gyro temperature sensitivity	±0.016%/°C
Compass scale	±1.3 Ga default. Up to ±8.1 Ga available
Compass resolution	12 bit
Compass sensitivity	5 mGa/digit
Compass non-linearity	0.1% full-scale

*Specifications subject to chang

Dongle	
Part number	TSS-DNG (Wireless Communication Dongle)
Dimensions	22.5mm x 65.6mm x 15mm (0.86 x 2.58 x 0.59 in.)
Weight	12 grams (0.42 oz)
Supply voltage	+5v USB
Communication interfaces	USB 2.0, 2.4GHz DSSS Wireless (FCC certified)
Wireless communication range	up to 200'
Wireless sensors supported	15 simultaneous
Wireless PAN Ids selectable	65536
Wireless channels selectable	16 (2.4GHz channel 11 through 26)
Processor	32-bit RISC running @ 60MHz

*Specifications subject to change



2.7 Physical Dimensions

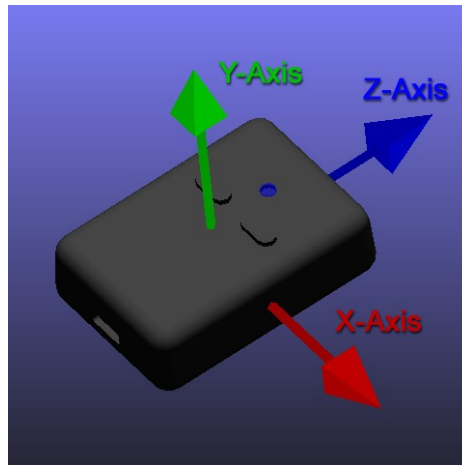
2.8 Axis Assignment

All YEI 3-Space Sensor product family members have re-mappable axis assignments and axis directions. This flexibility allows axis assignment and axis direction to match the desired end-use requirements.

The natural axes of the 3-Space Sensor are as follows:

- The positive X-axis points out of the right hand side of the sensor, which is the side that is facing right when the buttons face upward and plug faces towards you.
- The positive Y-axis points out of the top of the sensor, the side with the buttons.
- The positive Z-axis points out of the front of the sensor, the side opposite the plug.

The natural axes are illustrated in the diagram below



Bear in mind the difference between natural axes and the axes that are used in protocol data. While they are by default the same, they can be remapped so that, for example, data axis Y could contain data from natural axis X. This allows users to work with data in a reference frame they are familiar with.

2.9 Wireless Terminology

The following provides a list of commonly used wireless concepts and their definitions.

Pan ID – Refers to a 16-bit number that can be assigned to each individual wireless unit or dongle. The pan ID serves the purpose of separating units into clusters or networks, such that a unit with one pan ID cannot communicate with a unit on another pan ID.

Channel – Refers to the frequency on which a given unit transmits or receives upon. There are 16 available channels, ranging from 11-26, inclusive. Certain channels may be more well-suited for wireless communication than others at any given time. Refer to the command listing to find out how to scan channels. Like the pan ID, units with different channels cannot communicate with each other.

Address – Each unit has a unique built-in and unchangeable address (also referred to as hardware ID), which can be found etched into the back of wireless units (but not dongles). When communicating with a unit, addresses are not used directly. Instead, a mapping is provided in the form of logical IDs.

Logical ID – Since direct addresses are cumbersome, these are provided as a means to easily communicate with a given unit. There are 15 such logical IDs. Each logical ID can be mapped to a hardware address to ease communication. A table of logical IDs and their corresponding hardware addresses can be found inside the suite under the Dongle submenu, under Wireless Communication Settings... For more information on reading or setting logical IDs, please refer to the command chart.

2.10 Wireless LED Modes

Both the dongle and wireless unit have built-in LEDs that are meant to convey information about the state of the respective device. Each unit and dongle may also have a custom color that can be set. The wireless unit will display the following LED colors under the following circumstances:

- Upon receipt of a packet, the wireless unit will flash green temporarily. This will occur regardless of whether the wireless unit is plugged in or not.
- When the wireless unit is plugged in and charging, the sensor will flash orange every second.
- When the wireless unit is plugged in and fully charged, the sensor will flash green every second.
- When the wireless unit falls below a certain battery life level, it will flash red in increasingly quicker intervals. Note that this does not happen if the sensor is plugged in.
- Upon receipt of a packet, the dongle will flash green temporarily.
- If the dongle transmits a packet that does not reach its destination, the dongle will flash red temporarily.

Under all other circumstances, both devices will display the custom color that has been set. In addition to this default behavior, it is possible to set a static LED mode, in which the above functionality will be overridden. In this case, the LED will display only the custom color and nothing else. Please refer to the command chart for information on setting static LED mode.

3. Description of the 3-Space Sensor

3.1 Orientation Estimation

The primary purpose of the 3-Space Sensor is to estimate orientation. In order to understand how to handle this estimation and use it in a meaningful way, there are a few concepts about the sensor that should be understood. The following sections describe these concepts.

3.1.1 Component Sensors

The 3-Space Sensor estimates orientation by combining the data it gets from three types of sensors: a gyroscope, an accelerometer, and a compass. A few things you should know about each of these sensors:

- **Accelerometer:** This sensor measures the acceleration due to gravity, as well as any other accelerations that occur. Because of this, this sensor is at its best when the 3-Space Sensor is sitting still. Most jitter seen as the orientation of the sensor changes is due to shaking causing perturbations in the accelerometer readings. To account for this, by default, when the 3-Space Sensor is being moved, the gyroscope becomes more trusted (becomes a greater part of the orientation estimate), and the accelerometer becomes less trusted.
- **Gyroscope:** This sensor measures angular motion. It has no ability to give any absolute orientation information like the accelerometer or compass, and so is most useful for correcting the orientation during sensor motion. Its role during these times becomes vital, though, as the accelerometer readings can become unreliable during motion.
- **Compass:** This sensor measures magnetic direction. The readings from the compass and accelerometer are used together to form the absolute component of orientation, which is used to correct any short term changes the gyroscope makes. Its readings are much more stable than those of the accelerometer, but it can be adversely affected by any ferrous metal or magnetic objects. When the accelerometer is less trusted, the compass is treated in the same way so as to avoid updates to orientation based on partial absolute information.

3.1.2 Scale, Bias, and Cross-Axis Effect

The readings taken from each component sensor are not in a readily usable form. The compass and accelerometer readings are not unit vectors, and the gyroscope readings aren't yet in radians per second. To convert them to these forms, scale and bias must be taken into account. Scale is how much larger the range of data read from the component sensor is than the range of data should be when it is converted. For example, if the compass were to give readings in the range of -500 to 500 on the x axis, but we would like it to be in the range of -1 to 1, the scale would be 500. Bias is how far the center of the data readings is from 0. If another compass read from -200 to 900 on the x axis, the bias would be 350, and the scale would be 550. The last parameter used in turning this component sensor data into usable data is cross-axis effect. This is the tendency for a little bit of data on one axis of a sensor to get mixed up with the other two. This is an effect experienced by the accelerometer and compass. There are 6 numbers for each of these, one to indicate how much each axis is affected by each other axis. Values for these are generally in the range of 1 to 10%. These parameters are applied in the following order:

- 1) Bias is subtracted from each axis
- 2) The three axes are treated as a vector and multiplied by a matrix representing scale and cross-axis parameters

Factory calibration provides default values for these parameters for the accelerometer and compass, and users should probably never need to change these values. To determine these parameters for the gyroscope, you must calibrate it. Read the Quick Start guide or the 3-Space Suite manual for more information on how to do this.

3.1.3 Reference Vectors

In order to get an absolute estimation of orientation from the accelerometer and compass, the sensor needs a reference vector for each to compare to the data read from it. The most obvious choice for these are the standard direction of gravity (down) and the standard direction of magnetic force (north), respectively. However, the sensor does provide several different modes for determining which reference vector to use:

- **Single Manual:** Uses 2 reference vectors it is given as the reference vectors for the accelerometer and compass.
- **Single Auto:** When the sensor powers on or is put into this mode, it calculates gravity and north and uses those calculated vectors as the reference vectors.
- **Single Auto Continual:** The same as Single Auto, but the calculation happens constantly. This can account for some shifts in magnetic force due to nearby objects or change of location, and also can help to cope with the instability of the accelerometer.
- **Multiple:** Uses a set of reference vectors from which the best are picked each cycle to form a single, final reference vector. This mode has the ability to compensate for certain errors in the orientation. In this mode the sensor will have a slightly slower update rate, but will provide greater accuracy. For information on how to set up this mode, see the Quick Start guide or the 3-Space Suite manual.

3.1.4 Kalman Filter

The component sensor data and reference vectors are fed into a Kalman filter, which uses statistical techniques to optimally combine the data into a final orientation reading.

3.1.5 Reference Orientation/Taring

Given the results of the Kalman filter, the sensor can make a good estimation of orientation, but it will likely be offset from the actual orientation of the device by a constant angle until it has been given a reference orientation. This reference orientation tells the sensor where you would like its zero orientation to be. The sensor will always consider the zero orientation to be the orientation in which the plug is facing towards you and top(the side with buttons on it) facing up. The sensor must be given a reference orientation that represents the orientation of the sensor when it is in the position in which you consider the plug to be towards you and the buttons up. The act of giving it this reference orientation to the sensor is called taring, just as some scales have a tare button which can be pressed to tell the scale that nothing is on it and it should read zero. For instructions on doing this, refer to the Quick Start guide or 3-Space Suite manual.

3.1.6 Other Estimation Parameters

The 3-Space Sensor offers a few other parameters to filter the orientation estimate. Please note that these only affect the final orientation and not the readings of individual component sensors.

- **Oversampling:** Oversampling causes the sensor to take extra readings from each of the component sensors and average them before using them to estimate orientation. This can reduce noise, but also causes each cycle to take longer proportional to how many extra samples are being taken.
- **Running Average:** The final orientation estimate can be put through a running average, which will make the estimate smoother at the cost of introducing a small delay between physical motion and the sensor's estimation of that motion.
- **Rho Values:** As mentioned earlier, by default the accelerometer and compass are trusted less than the gyros when the sensor is in motion. Rho values are the mechanism that handles the concept of trust. They involve parameters, one for the accelerometer and one for the compass, that indicate how much these component sensors are to be trusted relative to the gyroscope. A lower value for the parameter means more trust. The default mode for this is “confidence mode”, where the rho value chooses between a minimum and maximum value based on how much the sensor is moving. The other option is to have a single, static rho value.

3.2 Communication

Obtaining data about orientation from the sensor or giving values for any of its settings is done through the sensor's communication protocol. The protocol can be used through either the USB port or wireless interface, using the 3-Space Wireless Dongle. A complete description of how to use this protocol is given in section 4 of this document. Also, you may instead use the 3-Space Suite, which provides a graphical method to do the same. To learn how to use this, read the 3-Space Suite manual.

3.3 Input Device Emulation

3.3.1 Axes and Buttons

The 3-Space Sensor has the ability to act as a joystick and/or mouse when plugged in through USB. Both of these are defined in the same way, as a collection of axes and buttons. Axes are input elements that can take on a range of values, whereas buttons can only either be on or off. On a joystick, the stick part would be represented as 2 axes, and all the physical buttons on it as buttons. The 3-Space Sensor has no physical joystick and only 2 physical buttons, so there are a number of options to use properties of the orientation data as axes and buttons. Each input device on the 3-Space Sensor has 2 axes and 8 buttons. For more information on setting these up, see the 3-Space Suite manual. All communication for these input devices is done through the standard USB HID(Human Interface Device) protocol.

3.3.2 Joystick

As far as a modern operating system is concerned, a joystick is any random collection of axes and buttons that isn't a mouse or keyboard. Joysticks are mostly used for games, but can also be used for simulation, robot controls, or other applications. The 3-Space Sensor, as a joystick, should appear just like any other joystick to an operating system that supports USB HID(which most do).

3.3.3 Mouse

When acting as a mouse, the 3-Space Sensor will take control of the system's mouse cursor, meaning if the mouse portion is not properly calibrated, using it could easily leave you in a situation in which you are unable to control the mouse cursor at all. In cases like this, unplugging the 3-Space Sensor will restore the mouse to normal operation, and unless the mouse enabled setting was saved to the sensor's memory, plugging it back in should restore normal operation. Using the default mouse settings, caution should be exercised in making sure the orientation estimate is properly calibrated before turning on the mouse. For help with this, see the Quick Start guide.

The mouse defaults to being in Absolute mode, which means that the data it gives is meant to represent a specific position on screen, rather than an offset from the last position. This can be changed to Relative mode, where the data represents an offset. In this mode, the data which would have indicated the edges of the screen in Absolute mode will now represent the mouse moving as quickly as it can in the direction of that edge of the screen. For more information, see command 251 in section 4.3.7, or the 3-Space Suite manual.

3.3.4 Wireless Joystick/Mouse

The 3-Space Dongle can be set up to receive joystick and mouse data from a 3-Space Sensor wirelessly and present this data to the computer via a USB interface. This is accomplished by supplying the logical ID of the wireless device that will act as the mouse/joystick. Commands 240 and 241 are used to enable the wireless joystick and mouse respectively. When either of these commands are invoked, the chosen wireless sensor will immediately begin transmitting the requested HID data to the dongle. The update rate at which this information is received is determined by command 215. Additionally, HID information may be sent synchronously or asynchronously from the wireless sensor to the dongle. Command 217 allows the user to set the desired mode. Synchronous HID mode is the default mode, in which the dongle automatically asks for the requested data first. This mode enjoys a high rate of reliability and it is quite easy to interlace regular protocol commands with HID data transmission/reception. This mode is slower, however, than asynchronous mode, since information must both be requested and received. Asynchronous mode, on the other hand, forces the sensor to automatically send HID information without being asked to do so by the dongle. This allows for much higher update rates, at the expense of reliability due to the increased number of wireless transmissions and potential collisions. It is recommended to use this mode only if you will be using the 3-Space Sensor only as an HID joystick or mouse at the given time.

3.4 Sensor Settings

3.4.1 Committing Settings

Changes made to the 3-Space Sensor will not be saved unless they are committed. This allows you to make changes to the sensor and easily revert it to its previous state by resetting the chip. For instructions on how to commit your changes, see the Quick Start guide or 3-Space Suite manual. Any changes relating to the multiple reference vector mode are an exception to this rule, as all these changes are saved immediately.

3.4.2 Committing Wireless Settings

In addition to committing sensor settings, there are also settings specific to wireless devices. In order to commit these settings, command 197 must be used. Note that committing the default settings will have no effect on wireless settings, while committing wireless settings will not change the default settings. A list of wireless settings for the sensor can be found in table 3.4.6 and a list of wireless settings for the dongle can be found in table 3.4.7.

3.4.3 Natural Axes

The natural axes of the 3-Space Sensor are as follows:

- The positive X-axis points out of the right hand side of the sensor, which is the side that is facing right when the buttons face upward and plug faces towards you.
- The positive Y-axis points out of the top of the sensor, the side with the buttons.
- The positive Z-axis points out of the front of the sensor, the side opposite the plug.

Bear in mind the difference between natural axes and the axes that are used in protocol data. While they are by default the same, they can be remapped so that, for example, data axis Y could contain data from natural axis X. This allows users to work with data in a reference frame they are familiar with. See section 2.8 for a diagram of the natural axes.

3.4.4 Sensor General Settings

Setting Name	Purpose	Default Value
Accelerometer Rho Value	Determine how trusted the accelerometer is	Confidence Mode, 5 to 100
Compass Rho Value	Determine how trusted the compass is	Confidence Mode, 5 to 100
Accelerometer Coefficients	Determines the scale, bias, and cross-axis parameters for the accelerometer	Factory calibrated
Compass Coefficients	Determines the scale, bias, and cross-axis parameters for the compass	Factory calibrated
Accelerometer Enabled	Determines whether the compass is enabled or not	TRUE
Compass Enabled	Determines whether the accelerometer is enabled or not	TRUE
Gyroscope Enabled	Determines whether the gyroscope is enabled or not	TRUE
Axis Directions	Determines what natural axis direction each data axis faces	+X, +Y, +Z
Sample Rate	Determines how many samples the sensor takes per cycle	1 from each component sensor
Running Average Percentage	Determines how heavy of a running average to run on the final orientation	0(no running average)
Desired Update Rate	Determines how long each cycle should take(ideally)	0 microseconds
Reference Mode	Determines how the accelerometer and compass reference vectors are determined	Single Auto
CPU Speed	Determines how fast the CPU will run	60 MHz
LED Color	Determines the RGB color of the LED	0,0,1(Blue)
LED Mode	Determines whether the LED mode is static or not.	0 (Non-static)
Joystick Enabled	Determines whether the joystick is enabled or not	TRUE
Mouse Enabled	Determines whether the mouse is enabled or not	FALSE
Button Gyro Disable Length	Determines how many cycles the gyro is ignored after a button is pressed	5
Multi Reference Weight Power	Determines what power each multi reference vector weight is raised to	10

Multi Reference Cell Divisions	Determines how many cells the multi reference lookup table is divided into per axis	4
Multi Reference Nearby Vectors	Determines how many nearby vectors each multi reference lookup table cell stores	8

3.4.5 Dongle General Settings

Setting Name	Purpose	Default Value
LED Color	Determines the RGB color of the LED	0,0,1(Blue)
Desired Update Rate	Determines how long each cycle should take (ideally)	0 microseconds
LED Mode	Determines whether the LED mode is static or not.	0 (Non-static)

3.4.6 Sensor Wireless Settings

Setting Name	Purpose	Default Value
PanID	Determines the panID of this sensor.	1
Address	Determines the address of this sensor.	Factory determined (cannot be set, only read)
Channel	Determines the channel of this sensor.	26

3.4.7 Dongle Wireless Settings

Setting Name	Purpose	Default Value
PanID	Determines the panID of this dongle.	1
Address	Determines the address of this dongle.	Factory determined (cannot be set, only read)
Channel	Determines the channel of this dongle.	26
Logical ID Table	Determines the mapping between logical ID and addresses.	Array of 15 unsigned 16-bit integers, values initialized to 0
Retries	Determines number of retries dongle will attempt on failed transaction	3
Joystick Logical ID	Determines the logical ID of the device that will act as the joystick, or -1 if there is no joystick desired.	-1
Mouse Logical ID	Determines the logical ID of the device that will act as the mouse, or -1 if there is no mouse desired.	-1
HID Update Rate	Update rate for requesting joystick/mouse information, in milliseconds.	15 (67 hz)
HID Asynchronous Mode	Determines whether joystick/mouse data transmission is asynchronous.	0
Asynchronous Flush Mode	Determines whether or not asynchronously requested data is automatically flushed or whether it must be requested via a dongle command.	1

4. 3-Space Sensor Usage/Protocol

4.1 Usage Overview

4.1.1 Protocol Overview

The 3-Space Sensor receives messages from the controlling system in the form of sequences of serial communication bytes called packets. For ease of use and flexibility of operation, two methods of encoding commands are provided: binary and text. Binary encoding is more compact, more efficient, and easier to access programmatically. ASCII text encoding is more verbose and less efficient yet is easier to read and easier to access via a traditional terminal interface. Both binary and ASCII text encoding methods share an identical command structure and support the entire 3-Space command set.

The 3-Space Sensor buffers the incoming command stream and will only take an action once the entire packet has been received and the checksum has been verified as correct(ASCII mode commands do not use checksums for convenience). Incomplete packets and packets with incorrect checksums will be ignored. This allows the controlling system to send command data at leisure without loss of functionality. The command buffer will, however, be cleared whenever the 3-Space Sensor is either reset or powered off/on.

Specific details of the 3-Space Sensor protocol and its control commands are discussed in the following pages.

4.1.2 Computer Interfacing Overview(USB)

When interfacing with a computer through USB, the 3-Space Sensor presents itself as a COM port, which provides a serial interface through which host may communication with the sensor unit by using protocol messages. The name of this COM port is specific to the operating system being used. It is possible to use multiple 3-Space Sensors on a single computer. Each will be assigned its own COM port. The easiest way to find out which COM port belongs to a certain sensor is to take note of what COM port appears when that sensor is plugged in(provided the drivers have been installed on that computer already. Otherwise, find out what COM port appears once driver installation has finished.) Additionally, each sensor can be identified programatically by reading the serial number of each attached sensor. For more information on how to install the sensor software on a computer and begin using it, see the Quick Start guide.

4.1.3 Computer Interfacing Overview(Wireless)

To interface to a sensor through a computer wirelessly, the 3-Space Dongle must be connected to the computer through USB. The Dongle will present itself as a COM port just as the 3-Space Sensor does. Each dongle can be associated with up to 15 wireless sensor units. To associate a sensor unit with a dongle, the user must place the desired sensor's serial number in one of the dongle's 15 logical wireless table slots. Any wireless 3-Space Sensors in range that have been given an address slot on the Dongle may then be communicated to using the Dongle. For information on how to set up the Dongle's address slots, see the Quick Start guide or <Dongle slot command ##>. For information on what data to send to the Dongle to communicate with a particular sensor, see section 4.3. The wireless communication protocol and wired communication protocol support the same commands, but are not identical. This allows the wireless protocol to include features that are specific to the nature of wireless communication such as wireless addressing, wireless reliability, and packet-loss handling, etc. For more information pertaining to the wired and wireless communication protocols, see sections 4.2 and 4.3 respectively.

4.2 Wired Protocol Packet Format

4.2.1 Binary Packet Format

The binary packet size can be three or more bytes long, depending upon the nature of the command being sent to the controller. Each packet consists of an initial “**start of packet**” byte, followed by a “**command value**” specifier byte, followed by zero or more “**command data**” bytes, and terminated by a packet “**checksum value**” byte.

Each binary packet is at least 3 bytes in length and is formatted as shown in figure 1

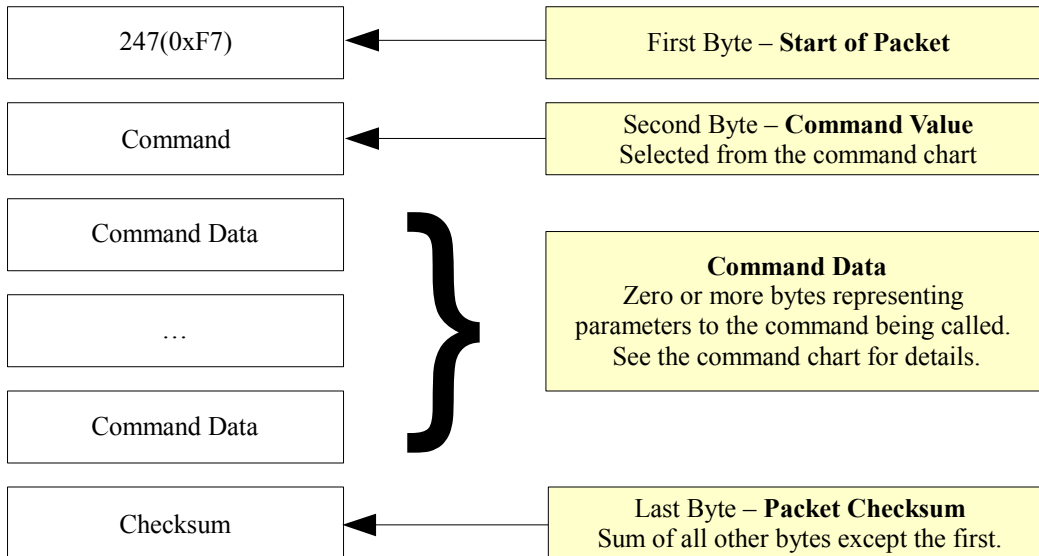


Figure 1 - Binary Command Packet Format

Binary Return Values:

When a 3 Space Sensor command is called in binary mode, any data it returns will also be in binary format. For example, if a floating point number is returned, it will be returned as its 4 byte binary representation.

For information on the floating point format, go here: http://en.wikipedia.org/wiki/Single_precision_floating_point_format

Also keep in mind that integer and floating point values coming from the sensor are stored in big-endian format.

The Checksum Value:

The checksum is computed as an arithmetic summation of all of the characters in the packet (except the checksum value itself) modulus 256. This gives a resulting checksum in the range 0 to 255. The checksum for binary packets is transmitted as a single 8-bit byte value.

4.2.2 ASCII Text Packet Format

ASCII text command packets are similar to binary command packets, but are received as a single formatted line of text. Each text line consists of the following: an ASCII colon character followed by an integral command id in decimal, followed by a list of ASCII encoded floating-point command values, followed by a terminating newline character. The command id and command values are given in decimal. The ASCII encoded command values must be separated by an ASCII comma character or an ASCII space character. Thus, legal command characters are: the colon, the comma, the period, the digits 0 through 9, the minus sign, the new-line, the space, and the backspace. When a command calls for an integer or byte sized parameter, the floating point number given for that parameter will be interpreted as being the appropriate data type. For simplicity, the ASCII encoded commands follow the same format as the binary encoded commands, but ASCII text encodings of values are used rather than raw binary encodings.

Each ASCII packet is formatted as shown in figure 2.

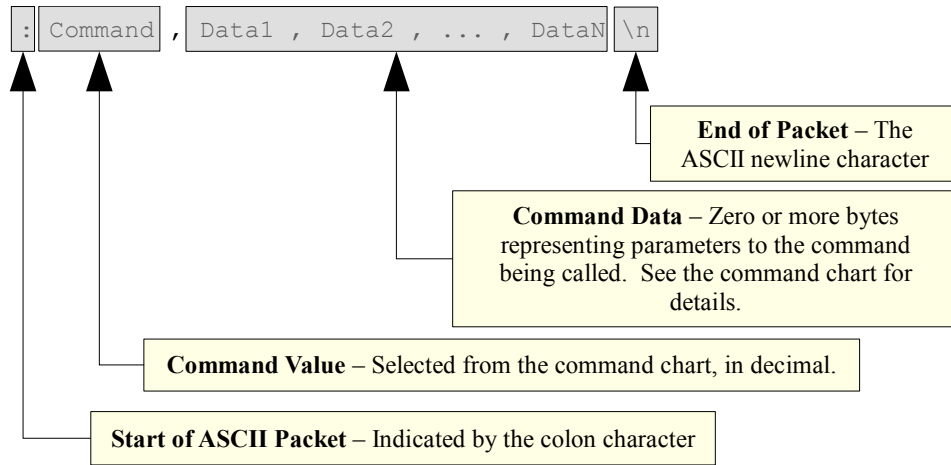


Figure 2 - ASCII Command Packet Format

Thus the ASCII packet consists of the the following characters:

- `:` – the ASCII colon character signifies the start of an ASCII text packet.
- `,` – the ASCII comma character acts as a value delimiter when multiple values are specified.
- `.` – the ASCII period character is used in floating point numbers.
- `0-9` – the ASCII digits are used to in integer and floating point values.
- `-` – the ASCII minus sign is used to indicate a negative number
- `\n` – the ASCII newline character is used to signify the end of an ASCII command packet.
- `\b` – the ASCII backspace character can be used to backup through the partially completed line to correct errors.

If a command is given in ASCII mode but does not have the right number of parameters, the entire command will be ignored. Also note that when communicating with the dongle or sensor in the 3-Space Suite, the newline is automatically appended to the input, thus it is not necessary to add it.

Sample ASCII commands:

<code>:0\n</code>	(If connected to the sensor)	Read orientation as a quaternion
<code>:106,2\n</code>	(If connected to the sensor)	Set oversample rate to 2
<code>:214\n</code>	(If connected to the dongle)	Read signal strength of most recent dongle reception
<code>:208,5\n</code>	(If connected to the dongle)	Read the hardware ID/address of the unit mapped to logical ID 5

ASCII Response:

All values are returned in ASCII text format when an ASCII-format command is issued. To read the return data, simply read data from the sensor until a Windows newline(a carriage return and a line feed) is encountered.

4.3 Wireless Protocol Packet Format

4.3.1 Wireless Communication Format

The protocol for communicating with sensors wirelessly is very similar to the wired protocol, but includes accommodations for wireless unit addressing and wireless communication failures. Thus, all wireless communication messages now also include an address specifying which sensor they are to be sent to. Additionally, each wireless protocol command returns status information pertaining to the success or failure of the wireless command. Any commands sent to address 254 will be sent to the dongle itself.

4.3.2 Binary Packet Format

The wireless binary packet format is very similar to the wired format. Each packet consists of an initial “**address**” byte, followed by a “**command value**” specifier byte, followed by zero or more “**command data**” bytes, and terminated by a packet “**checksum value**” byte.

Each wireless binary packet is at least 3 bytes in length and is formatted as shown in figure 3

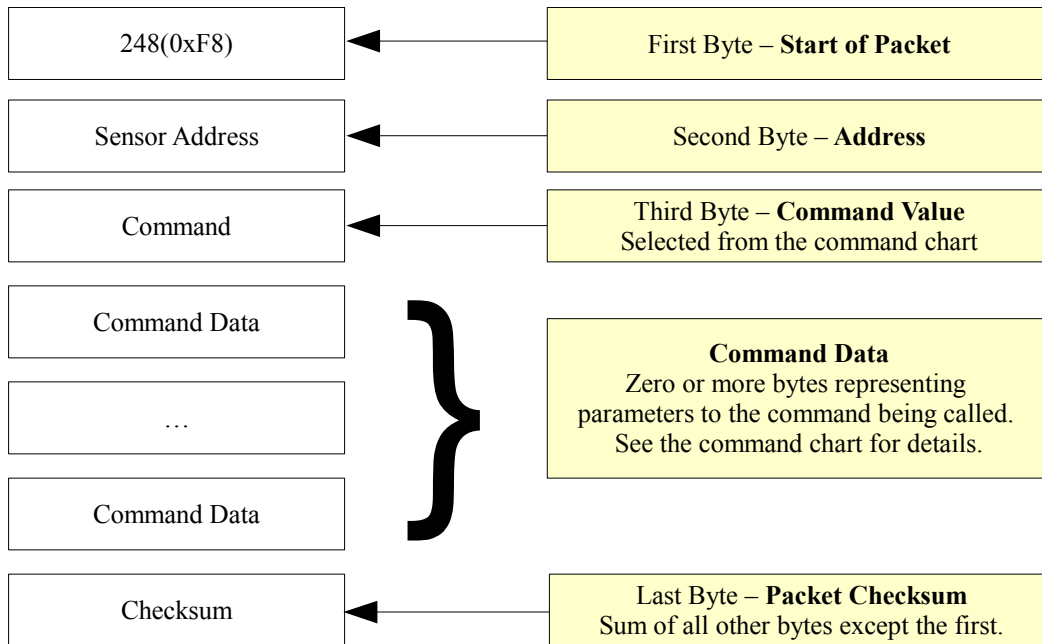
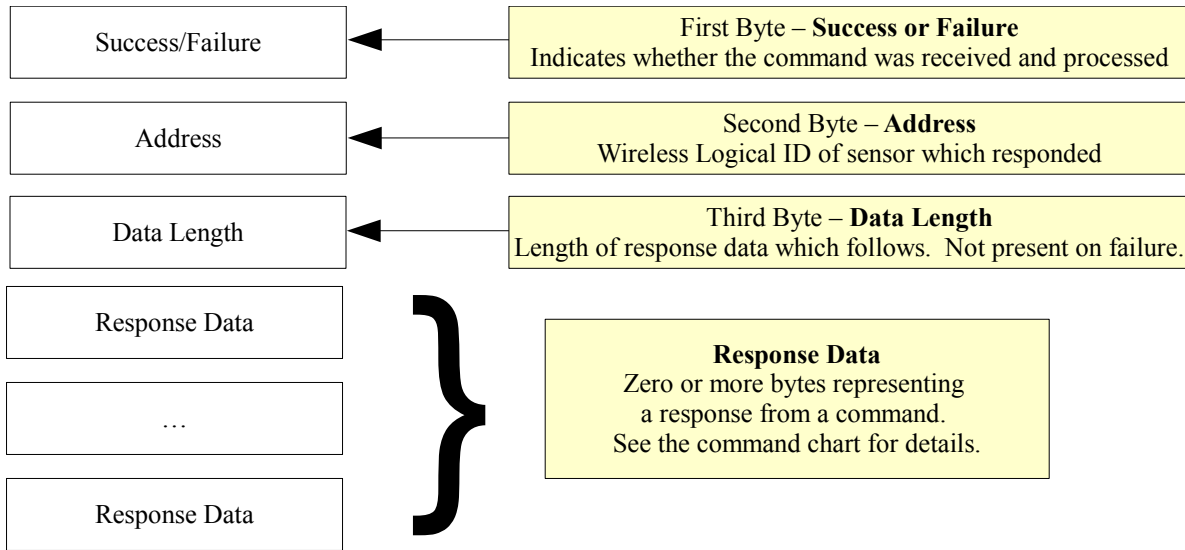


Figure 3 - Wireless Binary Command Packet Format

4.3.3 Binary Command Response



When a binary command is invoked wirelessly, before the data it would normally return in wired mode, it will return status bytes. First is the **success byte**, which is a 0 if the command was successful and non-0 if it was not. Some things which can cause a failure are:

- The lack of corresponding wireless sensor at the specified address.
- Wireless communication errors or dropped packets.
- Improper command formatting or data length

Second is the **address byte**. This indicates which sensor sent the response. If the success byte indicates a success, the third byte, the **data length byte**, will be present as well, indicating how much data follows it. All of the remaining data can be retrieved by reading the number of bytes indicated by **data length**. Keep in mind also that when the dongle is communicated to as if it were a wireless sensor, it will return data in the same format as a wireless response. Since all dongle commands that are formatted properly return success and the dongle address is a constant 254, all dongle responses begin with 00 FE.

4.3.4 Sample Binary Commands

Command	Description	Potential Response
F8 01 00 00	Read orientation as a quaternion from sensor 1	00 01 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3F 80 00 00
F8 05 6A 02 6C	Set oversample rate to 2 on sensor 5	00 05 00
F8 03 E6 E6	Read version string from sensor 3	00 03 0C 54 53 53 57 49 52 30 36 30 31 31 31
F8 0D EC EC	Read clock speed from sensor 13	00 0D 04 03 93 87 00
F8 09 77 00 00 00 00 BF 80 00 00 00 00 00 00	Set accelerometer reference vector to (0.0, -1.0, 0.0) on sensor 9	00 09 00
F8 FE C0 BE	Read the panID from the dongle	00 FE 02 00 01
F8 FE D7 14 E9	Set HID update rate to 20ms	00 FE 00

4.3.5 ASCII Text Packet Format

Wireless ASCII packets are very similar to wired ASCII packets. Each wireless ASCII packet is formatted as shown in figure 4.

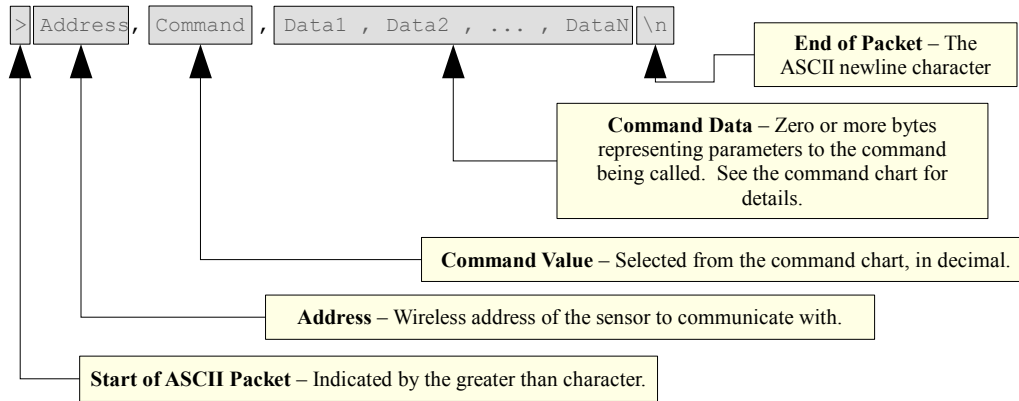


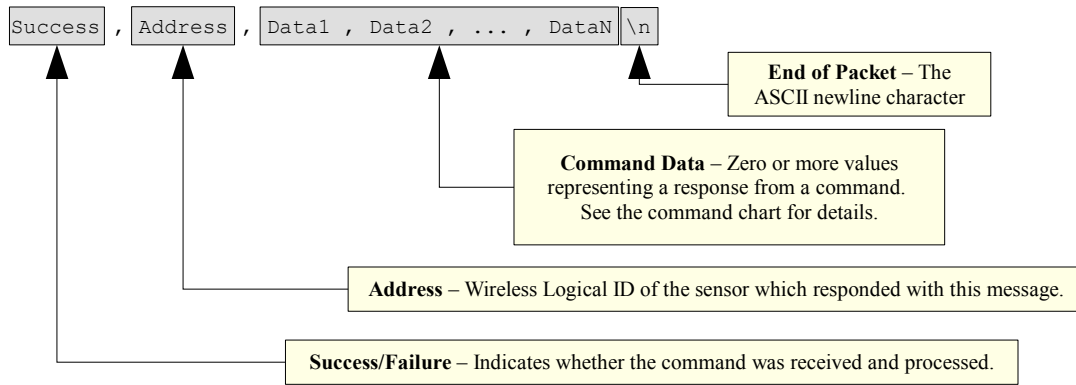
Figure 4 - Wireless ASCII Command Packet Format

Thus the ASCII packet consists of the the following characters:

- `>` – the ASCII greater than character signifies the start of an ASCII text packet.
- `,` – the ASCII comma character acts as a value delimiter when multiple values are specified.
- `.` – the ASCII period character is used in floating point numbers.
- `0~9` – the ASCII digits are used to in integer and floating point values.
- `-` – the ASCII minus sign is used to indicate a negative number
- `\n` – the ASCII newline character is used to signify the end of an ASCII command packet.
- `\b` – the ASCII backspace character can be used to backup through the partially completed line to correct errors.

If a command is given in ASCII mode but does not have the right number of parameters, the entire command will be ignored.

4.3.6 ASCII Command Response



When an ASCII command is called wirelessly, before the data it would normally return in wired mode, it will return status values, each separated by a comma. First is the **success/failure value**, which is a 0 if the command was successful and 1 if it was not. Some things which can cause a failure are:

- The lack of a sensor present wirelessly
- Communication interference causing the wireless sensor to not respond
- Improper command formatting or data length

Second is the **address**. This indicates which sensor sent the response. So for the command `:7,0\n`, the response might be `0,7,0.0,0.0,0.0,1.0\n`. Failures will only contain the status values and no data, so a failure of the above command would look like `1,7\n`.

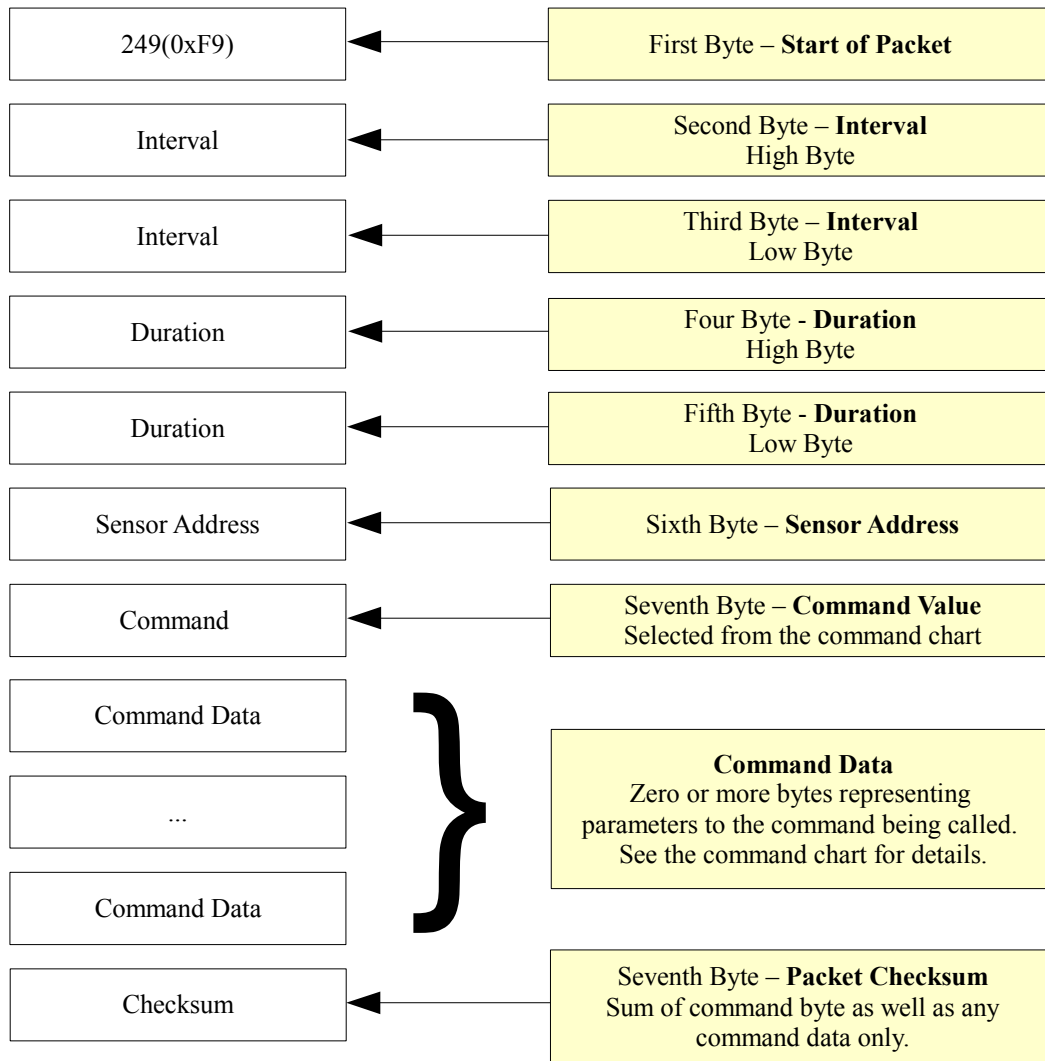
4.3.7 Sample ASCII Commands

Command	Description	Potential Response
<code>>0,1\n</code>	Read orientation as a quaternion from sensor 1	<code>0,1,0.0,0.0,0.707,0.707\n</code>
<code>>5,106,2\n</code>	Set oversample rate to 2 on sensor 5	<code>0,5\n</code>
<code>>3,230\n</code>	Read version string from sensor 3	<code>0,3,TSSWIR060111\n</code>
<code>>13,236\n</code>	Read clock speed from sensor 13	<code>0,13,60000000\n</code>
<code>>9,119,0.0,-1.0,0.0\n</code>	Set accelerometer reference vector to (0.0, -1.0, 0.0) on sensor 9	<code>0,9\n</code>
<code>>254,210\n</code>	Read wireless channel strengths from dongle	<code>0,254,0,4,0,0,0,4,0,1,0,2,0,1,4,0,1,0\n</code>
<code>>254,196,1\n</code>	Set LED mode to static on the dongle	<code>0,254\n</code>

4.4 Wireless Asynchronous Protocol

4.4.1 Asynchronous Communication Format

In addition to the standard request/response communication paradigm, the 3-Space sensor also supports an asynchronous protocol. All asynchronous requests are initiated through the dongle and sent to the corresponding sensor, but only once. This effectively halves communication overhead allowing a wireless sensor to automatically respond with given data at the specified interval for a specified amount of time. Providing that the command format is correct, the sensor given by the specified address will begin automatically transmitting the requested data upon receipt of the dongle's request. Asynchronous requests follow the same format as regular commands, in that the dongle will return status bytes indicating whether or not the asynchronous request was received by the sensor in question. Following is the general format for asynchronous requests:



4.4.2 Interval and Duration

Both the interval and duration are 16-bit unsigned integers representing, in milliseconds, how often data will be sent, and for how long data will be sent. An interval of zero can be specified, which will force the sensor to send the information as fast as possible. Specifying a 0xFFFF for the duration will result in an indefinite duration, while a duration of zero will terminate the asynchronous transmissions altogether for the requested sensor.

4.4.3 Starting/Stopping Asynchronous Transmissions

Since asynchronous requests return status bytes as a normal command would, it is possible to tell whether or not the asynchronous request was acknowledged. Just like any other command, the request itself will return a status code, the logical ID of the sensor, and the number of data bytes. Since these commands return no data, the number of data bytes will always be zero, and in the case of a failure, there will be no data byte present. Since there is presumably little wireless traffic occurring while the sensors are being placed into asynchronous mode, most asynchronous start requests will be received without issue. However, stopping asynchronous transmissions can be more difficult depending on the number of sensors communicating, since there will be more wireless collisions, more channel noise, and worse, the wireless sensor might even be in the middle of sending an asynchronous data transmission at the same time the dongle is requesting that it stop asynchronous communication. The best approach is to attempt each start and stop a number of times until a success is confirmed. If a failure is read, simply re-send the last asynchronous stop attempt. This can be done in a loop, talking to multiple sensors per iteration. This has the added effect that the remaining asynchronous transmissions become easier to stop the less of them there are.

4.4.4 Asynchronous Auto Flush

By default, once a dongle receives data that has been transmitted asynchronously, the data will be flushed as soon as it is received. For this reason, it is optimal to always be reading out data from the dongle once asynchronous mode has been initiated. The data format is nearly the same as a standard binary data response (see section 4.3.3). The only difference in the two is that the success byte will always read 0 in the case of asynchronous data, since there are no continuous requests, and no concept of failure from the dongle's perspective. Please note that asynchronous requests can only be invoked in a binary format, and that all asynchronous responses are binary data as well. For information on changing the asynchronous flush mode, please refer to dongle command 176 in the command chart.

4.4.5 Asynchronous Manual Flush

The dongle can also be configured to flush out asynchronous data only once requested. Additionally, it is possible to enable timestamps for the requested data, as well as prevent all data from certain sensors from being output by the dongle. Please refer to section 4.4.6 or dongle commands 178, 179, 180 and 181 in the command chart for more information. Once the asynchronous manual flush mode has been configured, there are two possible ways to retrieve the data. In the case that only one sensor is configured to transmit data asynchronously, a single read will most likely suffice. The single asynchronous read can be invoked by sending dongle command 182 followed by the logical ID of the sensor that has been configured to transmit asynchronously. The format for this returned data is slightly different than the auto-flushed data, in that there is no success byte. Instead, the dongle will output the logical ID of the sensor and the size of the requested data, followed by the data itself. If timestamps are enabled, this data size will include the size of the timestamp, which is a 32-bit unsigned integer. Note that it is possible that the data size can be zero. This indicates that the dongle has not received any data since the last time it was polled. Following is a snippet of pseudocode showing how to retrieve data with a single read:

```

Enable asynchronous communication for sensor with logical ID N.
Enable asynchronous manual flush mode.
Issue command 182 with single byte parameter N.
Read 1 byte, store as logicalID. (This should be the same as N)
Read 1 byte, store as dataSize.
If dataSize > 0:
    If timestamps are enabled:
        Read 4 bytes, store as timeStamp
        Read dataSize - 4 bytes, store as requestedData
    else:
        Read dataSize bytes, store as requestedData

```

In the case that the dongle has sent multiple asynchronous requests to multiple units, it is much more efficient to perform a bulk read, instead of calling the single read for each logical ID. The bulk read command can be invoked by issuing command 183. The format of this returned data varies slightly from single reads. The first two bytes that are read

will be the size of all of the returned data (the size is a 16-bit unsigned integer). From there, the next byte will be the logical ID of a sensor that has been enabled for flushing out data. The following byte will be the size of the data from that particular sensor, followed by the data itself. If timestamps are enabled, this data size will include the size of the timestamp, which is a 32-bit unsigned integer in units of milliseconds. Note that it is possible that the data size can be zero. Continue to loop through the remaining bytes until all data has been collected. Following is a snippet of pseudocode showing how to retrieve data with a bulk read:

```

Enable asynchronous communication for all desired sensors N.
Enable asynchronous manual flush mode.
Issue command 183.
Read two bytes, store as clusterSize (Total size of all data).
idx = 0
while idx < clusterSize:
    Read 1 byte, store as logicalID.
    idx += 1
    Read 1 byte, store as dataSize.
    idx += 1
    If dataSize > 0:
        If timestamps are enabled:
            Read 4 bytes, store as timeStamp
            Read dataSize - 4 bytes, store as requestedData
        else:
            Read dataSize bytes, store as requestedData
    idx += dataSize

```

4.4.6 Asynchronous Timestamps and Flush Bits

When requesting asynchronously-transmitted data in manual flush mode, it is possible to add timestamps to the output data, by using command 178—passing a parameter of 1 will enable timestamps, while 0 will disable them. Command 179 will return a value indicating whether or not timestamps are enabled. Timestamps are 32-bit unsigned integers representing an absolute time in microseconds. This provides for nearly 71 minutes of continuous timestamping until the value wraps around. Additionally, you can prevent the dongle from outputting data belonging to certain sensors. This is useful if you are bulk reading, and know with certainty, that you will only be communicating with a certain number of sensors, or a particular set of sensors that are already mapped in the address table in a specific manner. This can be set with command 180. For example, calling command 180 with a parameter of 0 and then another 0 will prevent the dongle from flushing any of the data unit 0 possibly sent—In this case, not even the logical ID or data length fields will be present in the bulk read. Calling the same command with a parameter of 1 will re-enable flushing for sensor 0. Command 181 followed by a parameter, will return the flush bit for the sensor specified by the parameter.

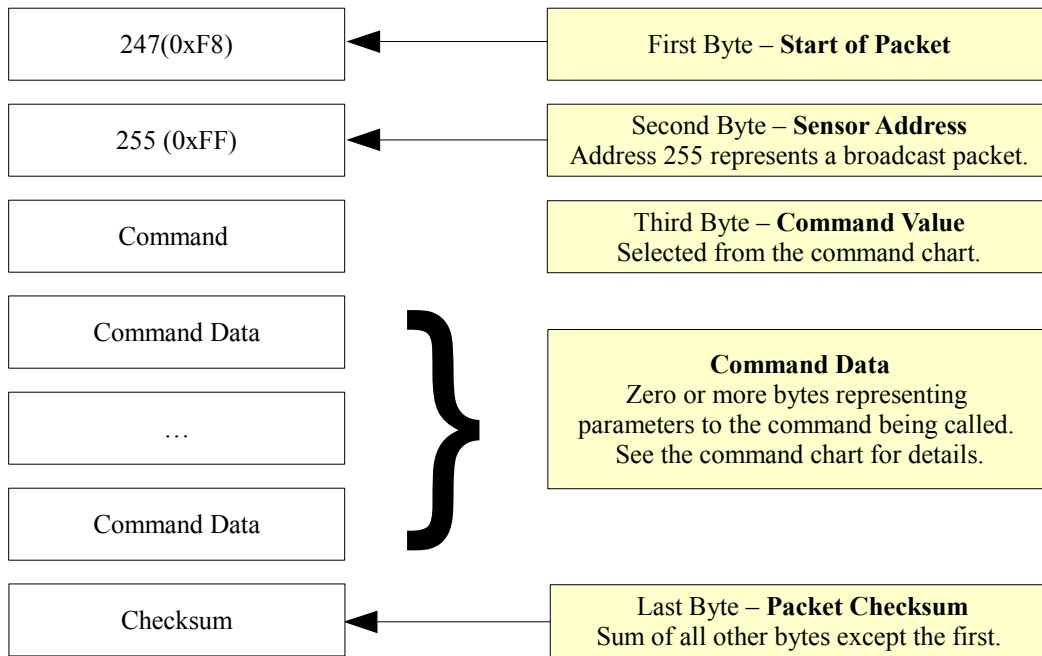
4.4.7 Sample Asynchronous Requests

Request	Description	Potential Response
F9 00 0F 00 64 03 00 03	Place sensor unit 3 into asynchronous transmission mode where it will run command 0 every 15 milliseconds, then transmit the resultant quaternion for 100 milliseconds.	00 03 00
F9 00 00 00 00 09 00 09	Disable asynchronous transmission mode for unit 9.	00 09 00
F9 00 00 FF FF 23 00 23	Attempt to send asynchronous request to non-existent unit 23.	01 23
F9 56 78 FF FF 05 20 25	Place sensor unit 5 into asynchronous transmission mode where it will send all raw sensor data every 22136 milliseconds for an indefinite period of time.	00 05 00

4.5 Wireless Broadcast Protocol

4.5.1 Broadcast Communication Format

Using the broadcast protocol, it is possible to send a message to every unit present on a particular channel. Note that due to the number of potential responses, potential collisions, and overall processing time involved in waiting for responses from all sensors, there are a few restrictions. First, only certain commands may be utilized in this way—namely, only commands that set state variables within the sensor. Thus, it is not possible to use commands that return data in this manner. As an example, you cannot broadcast an orientation request. You can, however, broadcast command 238 to change the LED color for all sensors. The other restriction is that it is not possible to tell whether a broadcast was successfully received. For this reason, it is recommended that broadcasts are sent multiple times. This will ensure that all expected sensors receive virtually all broadcast commands. Following is the general format for broadcasts:



4.5.2 Sample Broadcasts

Request	Description	Potential Response
F8 FF EE 63 80 00 00 00 00 00 00 00 00 00 00 D0	Set all sensor LEDs to red. (Checksum is (FF + EE + 63 + 80) mod 256.)	None
F8 FF E5 D3	Place all sensors into bootloader mode.	None

4.6 Command Overview

There are over 90 different command messages that are grouped numerically by function. Unused command message bytes are reserved for future expansion.

When looking at the following command message tables, note the following:

- The “Data Len” field indicates the number of additional data-bytes the command expects to follow the command-byte itself. This number doesn't include the Start of Packet, Command, or Checksum bytes. Thus, the total message size can be calculated by adding three bytes to the “Data Len” listed in the table.
- Likewise, the “Return Data Len” field indicates the number of data-bytes the command delivers back to the sender once the command has finished executing.
- Under “Return Data Details”, each command lists the sort of data which is being returned and next to this in parenthesis the form this data takes. For example, a quaternion is represented by 4 floating point numbers, so a command which returns a quaternion would list “Quaternion(float x4)” for its return data details.
- Command length information only applies to binary commands, as ascii commands can vary in length.
- For quaternions, data is always returned in x, y, z, w order.
- Euler angles are always returned in pitch, yaw, roll order.
- When calling commands in ASCII mode, there is no fixed byte length for the parameter data or return data, as the length depends on the ASCII encoding.

4.6.1 Commands for Reading Filtered Sensor Data

These commands return sensor data which has been filtered using a Kalman filter. None of these commands take any parameters, they only return data.

Command	Description	Return Len	Return Data Details
0(0x00)	Read filtered, tared orientation(Quaternion)	16	Quaternion(float x4)
1(0x01)	Read filtered, tared orientation(Euler Angles)	12	Euler Angles(float x3)
2(0x02)	Read filtered, tared orientation(Rotation Matrix)	36	Rotation Matrix(float x9)
3(0x03)	Read filtered, tared orientation(Axis Angle)	16	Axis(float x3), Angle(float)
4(0x04)	Read filtered, tared orientation(Two Vector(Forward, Down))	24	Vector(float x3), Vector(float x3)
5(0x05)	Read filtered gyro rates	16	Quaternion(float x4)
6(0x06)	Read filtered, untared orientation (Quaternion)	16	Quaternion(float x4)
7(0x07)	Read filtered, untared orientation (Euler Angles)	12	Euler Angles(float x3)
8(0x08)	Read filtered, untared orientation (Rotation Matrix)	36	Rotation Matrix(float x9)
9(0x09)	Read filtered, untared orientation (Axis Angle)	16	Axis(float x3), Angle(float)
10(0x0A)	Read filtered, untared orientation (Two Vector(Forward, Down))	24	Vector(float x3), Vector(float x3)
11(0x0B)	Read filtered, tared forward and down vectors in sensor reference frame (Two Vector(Forward, Down))	24	Vector(float x3), Vector(float x3)
12(0x0C)	Read filtered, North, Earth vectors in sensor reference frame(Two Vector(North, Earth))	24	Vector(float x3), Vector(float x3)

4.6.2 Commands for Reading Normalized Sensor Data

These commands return sensor data which has been converted from a raw form to a form that represents a real world quantity, but has not yet been used with the Kalman filter. None of these commands take any parameters, they only return data.

Command	Description	Return Len	Return Data Details
32(0x20)	Read all(gyro, accelerometer, compass)	36	Vector(float x3), Vector(float x3), Vector(float x3)
33(0x21)	Read gyros	12	Vector(float x3)
34(0x22)	Read accelerometer	12	Vector(float x3)
35(0x23)	Read compass	12	Vector(float x3)
36(0x24)	Read temperature C	4	float
37(0x25)	Read temperature F	4	float
38(0x26)	Read confidence factor	4	float

4.6.3 Commands for Reading Raw Sensor Data

These commands return sensor data just as it was when it was read from each sensor. None of these commands take any parameters, they only return data.

Command	Description	Return Len	Return Data Details	Data Len	Data Details
64(0x40)	Read all raw	36	Vector(float x3), Vector(float x3), Vector(float x3)	0	
65(0x41)	Read gyro raw	12	Vector(float x3)	1	Gyro range(byte: 0 for ± 250 dps, 1 for ± 500 dps, 2 for ± 2000 dps)
66(0x42)	Read accelerometer raw	12	Vector(float x3)	1	Accel range (byte: 0 for ± 2 g, 1 for ± 4 g, 2 for ± 8 g)
67(0x43)	Read compass raw	12	Vector(float x3)	1	Compass range (byte: 0 for ± 0.88 Ga, 1 for ± 1.3 Ga, 2 for ± 1.9 Ga, 3 for ± 2.5 Ga, 4 for ± 4.0 Ga, 5 for ± 4.7 Ga, 6 for ± 5.6 Ga, 7 for ± 8.1 Ga)

4.6.4 Commands for Setting Filter Parameters

These commands allow the configuration of parameters associated with the Kalman filter. Most of these commands take parameters, none return any data.

Command	Description	Data Len	Data Details
96(0x60)	Tare with current orientation	0	
97(0x61)	Tare with quaternion	16	Quaternion(float x4)
98(0x62)	Tare with rotation matrix	36	Rotation Matrix(float x9)
99(0x63)	Set static rho mode(Accelerometer)	4	Rho value(float)
100(0x64)	Set confidence rho mode(Accelerometer)	8	Min rho value(float), Max rho value(float)
101(0x65)	Set static rho mode(Compass)	4	Rho value(float)
102(0x66)	Set confidence rho mode(Compass)	8	Min rho value(float), Max rho value(float)
103(0x67)	Set desired update rate	4	Update rate in microseconds(int)
104(0x68)	Set multi reference vectors with current orientation	0	
105(0x69)	Set reference vector mode	1	Mode(byte, 0 for single static, 1 for single auto, 2 for single auto continuous, 3 for multi)
106(0x6a)	Set oversample rate	1	Rate(1 for off, 2+ for number of samples per frame)
107(0x6b)	Enable/disable gyros	1	Mode(byte, 0 for disabled, 1 for enabled)
108(0x6c)	Enable/disable accelerometer	1	Mode(byte, 0 for disabled, 1 for enabled)
109(0x6d)	Enable/disable compass	1	Mode(byte, 0 for disabled, 1 for enabled)
110(0x6e)	Reset multi reference vectors to zero	0	
111(0x6f)	Set multi reference resolution	2	Resolution(byte x2, number of cell divisions, number of nearby vectors)
112(0x70)	Set compass multi reference vector	13	Index(byte), Vector(float x3)
113(0x71)	Set compass multi reference check vector	13	Index(byte), Vector(float x3)
114(0x72)	Set accel multi reference vector	13	Index(byte), Vector(float x3)
115(0x73)	Set accel multi reference check vector	13	Index(byte), Vector(float x3)
116(0x74)	Set axis directions	1	Axis direction byte
117(0x75)	Set running average percent	4	Percent(float)
118(0x76)	Set compass reference vector	12	Vector(float x3)
119(0x77)	Set accelerometer reference vector	12	Vector(float x3)
120(0x78)	Reset Kalman filter	0	
121(0x79)	Set accelerometer range	1	Accel range(byte, 0 for 2G, 0x10 for 4G, 0x30 for 8G)
122(0x7a)	Set multi reference weight power	4	Weight power(float)
123(0x7b)	Enable / disable filter	1	Mode(byte, 0 for disabled, 1 for enabled)
124(0x7c)	Set running average mode	1	Mode(byte, 0 for normal, 1 for confidence)
125(0x7d)	Set gyroscope range	1	Gyro range mode(byte, 0 for 250 dps, 1 for 500 dps, 2 for 2000 dps)
126(0x7e)	Set compass range	1	Compass range mode(byte, see command details for options)

4.6.5 Commands for Reading Filter Parameters

These commands allow the reading of parameters associated with the Kalman filter. All these commands return data, and accept no parameters.

Command	Description	Return Len	Return Data Details	Data Len	Data Details
128(0x80)	Read tare orientation(Quaternion)	16	Quaternion(float x4)	0	
129(0x81)	Read tare orientation(Rotation Matrix)	36	Rotation Matrix(float x9)	0	
130(0x82)	Read rho data(Accelerometer)	9	Rho mode(byte, 1 for confidence, 0 for static), min/static rho(float), (max rho(float))	0	
131(0x83)	Read rho data(Compass)	9	Rho mode(byte, 1 for confidence, 0 for static), min/static rho(float), (max rho(float))	0	
132(0x84)	Read current update rate	4	Update rate in microseconds(int)	0	
133(0x85)	Read compass reference vector	12	Vector(float x3)	0	
134(0x86)	Read accelerometer reference vector	12	Vector(float x3)	0	
135(0x87)	Read reference vector mode	1	Mode(byte, 0 for single static, 1 for single auto, 2 for single auto continuous, 3 for multi)	0	
136(0x88)	Read compass multi reference vector	12	Vector(float x3)	1	Index
137(0x89)	Read compass multi reference check vector	12	Vector(float x3)	1	Index
138(0x8a)	Read accel multi reference vector	12	Vector(float x3)	1	Index
139(0x8b)	Read accel multi reference check vector	12	Vector(float x3)	1	Index
140(0x8c)	Read gyro enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
141(0x8d)	Read accelerometer enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
142(0x8e)	Read compass enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
143(0x8f)	Read axis directions	1	Axis direction byte	0	
144(0x90)	Read oversample rate	1	Rate(1 for off, 2+ for number of samples per frame)	0	
145(0x91)	Read running average percent	4	Percent(float)	0	
146(0x92)	Read desired update rate	4	Update rate in microseconds(int)	0	
147(0x93)	Read Kalman filter's covariance matrix	36	Covariance Matrix(float x9)	0	
148(0x94)	Read accelerometer range	1	Accel range(byte, 0 for 2G, 0x10 for 4G, 0x30 for 8G)	0	
149(0x95)	Read multi reference weight power	4	Weight power(float)	0	
150(0x96)	Read multi reference resolution	2	Resolution(byte x2, number of cell divisions, number of nearby vectors)	0	
151(0x97)	Read number of multi reference cells	4	Number of cells(int)	0	
152(0x98)	Read filter enable state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
153(0x99)	Read running average mode	1	Mode(byte, 0 for normal, 1 for confidence)	0	
154(0x9a)	Read gyroscope range	1	Gyro range mode(byte, 0 for 250 dps, 1 for 500 dps, 2 for 2000 dps)	0	
155(0x9b)	Read compass range	1	Gyro range mode(byte, see command details for options)	0	

4.6.6 Commands for Calibration

These commands allow the configuration and reading of calibration parameters and enabling of calibration modes.

Command	Description	Return Len	Return Data Details	Data Len	Data Details
160 (0xa0)	Set compass calibration parameters	0		48	Bias(float x3), Matrix(float x9)
161 (0xa1)	Set accelerometer calibration parameters	0		48	Bias(float x3), Matrix(float x9)
162 (0xa2)	Read compass calibration parameters	48	Bias(float x3), Matrix(float x9)	0	
163 (0xa3)	Read accelerometer calibration parameters	48	Bias(float x3), Matrix(float x9)	0	
164 (0xa4)	Read gyro calibration parameters	24	Bias(float x3), High range bias(float x3)	0	
165 (0xa5)	Begin gyro auto-calibration	0		0	
166(0xa6)	Set gyro calibration parameters	0		24	Bias(float x3), High range bias(float x3)
167(0xa7)	Set lookup-table vertex value	0		15	Value type(byte, 0 for compass, 1 for accelerometer), Index(short), Value(floatx3)
168(0xa8)	Read lookup-table vertex value	12	Value(floatx3)	3	Value type(byte, 0 for compass, 1 for accelerometer), index(short)

4.6.7 Commands for Wireless Sensor Units

These commands are applicable to wireless units. All, except battery status commands, are applicable to both dongle units and sensor units.

Command	Description	Return Data Len	Return Data Details	Data Len	Data Details
192(0xc0)	Read wireless panID	2	PanID (short)	0	
193(0xc1)	Set wireless panID	0		2	PanID (short)
194(0xc2)	Read wireless channel	1	Channel (byte)	0	
195(0xc3)	Set wireless channel	0		1	Channel (byte)
196(0xc4)	Set LED mode	0		1	Mode (byte)
197(0xc5)	Commit wireless settings	0		0	
198(0xc6)	Read wireless address	2	Address (short)	0	
200(0xc8)	Read LED mode	1	Mode (byte)	0	
201(0xc9)	Read battery voltage	4	Voltage (float)	0	
202(0xca)	Read battery percent remaining	2	Percent (short)	0	
203(0xcb)	Read battery status	1	Status (byte)	0	

4.6.8 Commands for Wireless Dongle Units

These commands are applicable only to wireless sensor units.

Command	Description	Return Data Len	Return Data Details	Data Len	Data Details
208(0xd0)	Read wireless association table entry ID	4	Hardware address (int)	1	Index (byte)
209(0xd1)	Set wireless association table entry ID	0		5	Index (byte), value (int)
210(0xd2)	Read wireless channel noise levels	16	Levels (byte x16)	0	
211(0xd3)	Set wireless retries	0		1	Retries (byte)
212(0xd4)	Read wireless retries	1	Retries (byte)	0	
213(0xd5)	Read wireless slots open	1	Slots (byte)	0	
214(0xd6)	Read signal strength	1	Reception strength (byte)	0	
215(0xd7)	Set HID update rate	0		1	Update interval in ms (byte)
216(0xd8)	Read HID update rate	1	Update interval in ms (byte)	0	

4.6.9 General Commands

These commands are for the configuration of the sensor as a whole as opposed to configuration of the filter or sensors. These apply to both the dongle and wireless sensor.

Command	Description	Return Data Len	Return Data Details	Data Len	Data Details
223(0xdf)	Read version extended	16	Version number(string)	0	
224(0xe0)	Restore factory settings	0		0	
225(0xe1)	Commit Settings	0		0	
226(0xe2)	Software reset	0		0	
227(0xe3)	Enable watchdog timer	0		4	Timeout rate in microseconds(int)
228(0xe4)	Disable watchdog timer	0		0	
229(0xe5)	Enter firmware update mode	0		0	
230(0xe6)	Get version	12	Version number(string)	0	
233(0xe9)	Set USB mode	0		1	Mode(byte, 2 for Unique, 1 for FTDI, 0 for CDC)
234(0xea)	Get USB mode	1	Mode(byte, 2 for Unique, 1 for FTDI, 0 for CDC)	0	
235(0xeb)	Set clock speed	0		4	Clock speed(int, Hz)
236(0xec)	Get clock speed	4	Clock speed(int, Hz)	0	
237(0xed)	Get serial number	4	Serial number(int)	0	
238(0xee)	Set LED color	0		12	Red(float), Green(float), Blue(float)
239(0xef)	Get LED color	12	Red(float), Green(float), Blue(float)	0	
251(0xfb)	Set mouse absolute/relative	0		1	Mode(0 for absolute, 1 for relative)
252(0xfc)	Read mouse absolute/relative	1	Mode(0 for absolute, 1 for relative)	0	
253(0xfd)	Set joystick and mouse present/removed	0		2	Mode(byte for each, 0 for removed, 1 for present)
254(0xfe)	Read joystick and mouse present/removed	2	Mode(byte for each, 0 for removed, 1 for present)	0	

4.6.10 HID Commands

These commands configure HID output. These commands are applicable only to wireless sensors.

Command	Description	Return Data Len	Return Data Details	Data Len	Data Details
240(0xf0)	Enable/Disable joystick	0		1	Mode(byte, 0 for disabled, 1 for enabled)
241(0xf1)	Enable/Disable mouse	0		1	Mode(byte, 0 for disabled, 1 for enabled)
242(0xf2)	Read joystick enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
243(0xf3)	Read mouse enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
244(0xf4)	Set control mode	0		3	Control class(byte), control index(byte), handler index(byte)
245(0xf5)	Set control data	0		7	Control class(byte), control index(byte), data point index(byte), data point(float)
246(0xf6)	Read control mode	1	Handler index(byte)	2	Control class(byte), control index(byte)
247(0xf7)	Read control data	4	Data point(float)	3	Control class(byte), control index(byte), data point index(byte)
248(0xf8)	Set button gyro disable length	0		1	Length in frames(byte)
249(0xf9)	Read button gyro disable length	1	Length in frames(byte)	0	
250(0xfa)	Read button state	1	Button state(byte, 1 bit per button)	0	

4.7 Command Details

In the tables below you'll find a description of each of the 3-Space Sensor commands and a brief explanation of how and where each command would be used.

Function:	Read filtered, tared orientation(Quaternion)
Command Value:	0 (0x00)
Return Data Bytes:	16
Return Data Format:	Quaternion(float x4)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as a quaternion.

Function:	Read filtered, tared orientation(Euler Angles)
Command Value:	1 (0x01)
Return Data Bytes:	12
Return Data Format:	Euler Angles(float x3)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as Euler angles.

Function:	Read filtered, tared orientation(Rotation Matrix)
Command Value:	2 (0x02)
Return Data Bytes:	36
Return Data Format:	Rotation Matrix(float x9)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as a quaternion.

Function:	Read filtered, tared orientation(Euler Angles)
Command Value:	3 (0x03)
Return Data Bytes:	16
Return Data Format:	Axis(float x3), Angle(float)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as an axis and an angle.

Function:	Read filtered, tared orientation(Two Vector(Forward, Down))
Command Value:	4 (0x04)
Return Data Bytes:	24
Return Data Format:	Vector(float x3), Vector(float x3)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as the forward and down vectors of the coordinate system defined by the orientation.

Function:	Read filtered gyro rates
Command Value:	5 (0x05)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the gyro rates as determined by taking the difference between the current orientation and the previous one.

User's Manual

Function:	Read filtered, untared orientation(Quaternion)
Command Value:	6 (0x06)
Return Data Bytes:	16
Return Data Format:	Quaternion(float x4)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as a quaternion.

Function:	Read filtered, untared orientation(Euler Angles)
Command Value:	7 (0x07)
Return Data Bytes:	12
Return Data Format:	Euler Angles(float x3)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as Euler angles.

Function:	Read filtered, untared orientation(Rotation Matrix)
Command Value:	8 (0x08)
Return Data Bytes:	36
Return Data Format:	Rotation Matrix(float x9)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as a quaternion.

Function:	Read filtered, untared orientation(Euler Angles)
Command Value:	9 (0x09)
Return Data Bytes:	16
Return Data Format:	Axis(float x3), Angle(float)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as an axis and an angle.

Function:	Read filtered, untared orientation(Two Vector(Forward, Down))
Command Value:	10 (0x0A)
Return Data Bytes:	24
Return Data Format:	Vector(float x3), Vector(float x3)
Description:	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as the forward and down vectors of the coordinate system defined by the orientation.

Function:	Read filtered, tared forward and down vectors in sensor reference frame (Two Vector(Forward, Down))
Command Value:	11 (0x0B)
Return Data Bytes:	24
Return Data Format:	Vector(float x3), Vector(float x3)
Description:	Returns the forward and down vectors as determined by the Kalman filter, relative to the tare orientation. This version returns the data as the forward and down vectors of the coordinate system defined by the sensor reference frame.

Function:	Read filtered, North, Earth vectors in sensor reference frame(Two Vector(North, Earth))
Command Value:	12(0x0C)
Return Data Bytes:	24
Return Data Format:	Vector(float x3), Vector(float x3)
Description:	Returns the North and Earth vectors as determined by the Kalman filter, relative to the global sensor references. Returned vectors are in the coordinate system defined by the sensor reference frame.

User's Manual

Function:	Read all(gyro, accelerometer, compass)
Command Value:	32 (0x20)
Return Data Bytes:	36
Return Data Format:	Vector(float x3), Vector(float x3), Vector(float x3)
Description:	Returns the normalized gyro, accelerometer, and compass values.

Function:	Read gyros
Command Value:	33 (0x21)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the normalized gyro rates.

Function:	Read accelerometers
Command Value:	34 (0x22)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the normalized gravity vector.

Function:	Read compass
Command Value:	35 (0x23)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the normalized north vector.

Function:	Read temperature C
Command Value:	36 (0x24)
Return Data Bytes:	4
Return Data Format:	float
Description:	Returns the temperature of the sensor in Celsius

Function:	Read temperature F
Command Value:	37(0x25)
Return Data Bytes:	4
Return Data Format:	float
Description:	Returns the temperature of the sensor in Fahrenheit

Function:	Read confidence factor
Command Value:	38 (0x26)
Return Data Bytes:	4
Return Data Format:	float
Description:	Returns a value indicating how much the compass and accelerometer are to be trusted to be unit vectors pointing in the proper directions at the moment. This value ranges from 0 to 1, with 1 being fully trusted and 0 being not trusted at all. This will change based on if the sensor is being moved and if it is near a strong magnetic field.

User's Manual

Function:	Read all raw(gyro, accelerometer, compass)
Command Value:	64 (0x40)
Return Data Bytes:	36
Return Data Format:	Vector(float x3), Vector(float x3), Vector(float x3)
Description:	Returns the raw gyro, accelerometer, and compass values. Values are according to the currently selected range for each respective sensor.

Function:	Read gyro raw
Command Value:	65 (0x41)
Data Bytes:	1
Data Format:	Gyro range (Byte, 0 for ± 250 dps, 1 for ± 500 dps, 2 for ± 2000 dps)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the raw gyro values in the specified measurement range.

Function:	Read accelerometer raw
Command Value:	66 (0x42)
Data Bytes:	1
Data Format:	Accelerometer range (Byte, 0 for ± 2 g, 1 for ± 4 g, 2 for ± 8 g)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the raw accelerometer values in the specified measurement range.

Function:	Read compass raw
Command Value:	67 (0x43)
Data Bytes:	1
Data Format:	Compass range (byte: 0 for ± 0.88 Ga, 1 for ± 1.3 Ga, 2 for ± 1.9 Ga, 3 for ± 2.5 Ga, 4 for ± 4.0 Ga, 5 for ± 4.7 Ga, 6 for ± 5.6 Ga, 7 for ± 8.1 Ga)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the raw compass values in the specified measurement range.

Function:	Tare with current orientation
Command Value:	96 (0x60)
Description:	Sets current filtered orientation as the tare orientation.

Function:	Tare with quaternion
Command Value:	97 (0x61)
Data Bytes:	16
Data Format:	Quaternion(float x4)
Description:	Sets the tare orientation to the orientation specified by the given quaternion.

Function:	Tare with rotation matrix
Command Value:	98 (0x62)
Data Bytes:	36
Data Format:	Rotation Matrix(float x9)
Description:	Sets the tare orientation to the orientation specified by the given rotation matrix.

User's Manual

Function:	Set static rho mode(Accelerometer)
Command Value:	99 (0x63)
Data Bytes:	4
Data Format:	Rho value(float)
Description:	Sets the rho mode for the accelerometer to static, using the given value as rho.

Function:	Set confidence rho mode(Accelerometer)
Command Value:	100 (0x64)
Data Bytes:	8
Data Format:	Min rho value(float), Max rho value(float)
Description:	Sets the rho mode for the accelerometer to confidence, using the given values as the min and the max. The confidence factor will be used in real time to interpolate between these to determine what rho value is used.

Function:	Set static rho mode(Compass)
Command Value:	101 (0x65)
Data Bytes:	4
Data Format:	Rho value(float)
Description:	Sets the rho mode for the compass to static, using the given value as rho.

Function:	Set confidence rho mode(Compass)
Command Value:	102 (0x66)
Data Bytes:	8
Data Format:	Min rho value(float), Max rho value(float)
Description:	Sets the rho mode for the compass to confidence, using the given values as the min and the max. The confidence factor will be used in real time to interpolate between these to determine what rho value is used.

Function:	Set desired update rate
Command Value:	103 (0x67)
Data Bytes:	4
Data Format:	Update rate in microseconds(int)
Description:	Sets the target update rate to the given rate. If the filter takes less time to update during a given frame than this rate specifies, the frame will be padded out to take the specified amount of time. If the filter takes more time to update than this rate, this rate will be ignored.

Function:	Set multi reference vectors with current orientation
Command Value:	104 (0x68)
Data Bytes:	0
Description:	Uses the current tared orientation to set up the reference vector for the nearest orthogonal orientation.

User's Manual

Function:	Set reference vector mode
Command Value:	105 (0x69)
Data Bytes:	1
Data Format:	Mode(byte, 0 for single static, 1 for single auto, 2 for single auto continuous, 3 for multi)
Description:	<p>Sets reference vector mode.</p> <p>-Single static mode uses a certain reference vector for the compass and another certain reference vector for the accelerometer at all times.</p> <p>-Single auto mode uses (0,-1,0) as the reference vector for the accelerometer at all times and uses the current average angle between the accelerometer and compass to calculate the compass reference vector. After that this mode acts like single static mode.</p> <p>-Single auto continuous mode uses (0,-1,0) as the reference vector for the accelerometer at all times and uses the average angle between the accelerometer and compass to constantly redetermine the compass reference vector.</p> <p>-Multi mode uses a collection of reference vectors for the compass and accelerometer, and selects which ones to use before each step of the filter.</p>

Function:	Set oversample rate
Command Value:	106 (0x6a)
Data Bytes:	1
Data Format:	Rate(1 for off, 2+ for number of samples per frame)
Description:	Sets the number of times to sample data for each run of the filter.

Function:	Enable/disable gyros
Command Value:	107 (0x6b)
Data Bytes:	1
Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Enables or disables the gyros(will be replaced with a still gyro reading if disabled).

Function:	Enable/disable accelerometer
Command Value:	108 (0x6c)
Data Bytes:	1
Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Enables or disables the accelerometer(This filter will not use this data if disabled).

Function:	Enable/disable compass
Command Value:	109 (0x6d)
Data Bytes:	1
Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Enables or disables the compass(This filter will not use this data if disabled).

Function:	Reset multi reference vectors to zero
Command Value:	110 (0x6e)
Data Bytes:	0
Description:	Resets all reference vectors in the multi reference scheme to zero.

User's Manual

Function:	Set multi reference resolution
Command Value:	111 (0x6f)
Data Bytes:	2
Data Format:	Mode(byte x2, number of cell divisions, number of nearby vectors)
Description:	<p>Sets number of cell divisions and number of nearby vectors per cell for the multi reference lookup table.</p> <p>By default, the multiple reference vector mode only deals with orientations obtainable by successive rotations of 90 degrees about any of the three axes. This command can adjust it to accept orientations obtainable by 45 degree rotations. For 90 degrees, give a "number of cell divisions" of 4, and for 45 give 8. In addition, the number of vectors near to any given orientation which the scheme will check can be adjusted as well. If this value is 0, only the nearest orientation will be checked. The largest this value can be is 32. Also note that the larger this value is, the longer the multiple reference vector mode will take to run each cycle.</p>

Function:	Set compass multi-reference vector
Command Value:	112 (0x70)
Data Bytes:	13
Data Format:	Index(byte), Vector(float x3)
Description:	Directly set compass multi reference vector at the specified index to the specified vector.

Function:	Set compass multi-reference check vector
Command Value:	113 (0x71)
Data Bytes:	13
Data Format:	Index(byte), Vector(float x3)
Description:	Directly set compass multi reference check vector at the specified index to the specified vector.

Function:	Set accel multi-reference vector
Command Value:	114 (0x72)
Data Bytes:	13
Data Format:	Index(byte), Vector(float x3)
Description:	Directly set accel multi reference vector at the specified index to the specified vector.

Function:	Set accel multi-reference check vector
Command Value:	115 (0x73)
Data Bytes:	13
Data Format:	Index(byte), Vector(float x3)
Description:	Directly set accel multi reference check vector at the specified index to the specified vector.

Function:	Set axis directions
Command Value:	116 (0x74)
Data Bytes:	1
Data Format:	Axis direction byte
Description:	<p>Set which directions each of the natural axes of the sensor point. The lower 3 bits are used to specify which axis each of the natural axes will read as: 000: XYZ(standard operation) 001: XZY 002: YXZ 003: YZX 004: ZXY 005: ZYX (For example, using ZXY means that whatever value appears as X on the natural axes will now be the Z component of any new data) The 3 bits above those are used to indicate which axes should be reversed. If it is cleared, the axis is pointing in the positive direction, and if it is set the axis is flipped. (Note: these are applied to the axes <i>after</i> the above conversion takes place) Bit 4: Positive/Negative Z (third resulting component) Bit 5: Positive/Negative Y (second resulting component) Bit 6: Positive/Negative X (first resulting component)</p>

Function:	Set running average percent
Command Value:	117 (0x75)
Data Bytes:	4
Data Format:	Percent(float)
Description:	<p>Sets what percentage of running average to use on the sensor's orientation. This is computed as follows: $total_orient = total_orient * percent$ $total_orient = total_orient + current_orient * (1 - percent)$ $current_orient = total_orient$ If the percentage is 0, the running average will be shut off completely.</p>

Function:	Set compass reference vector
Command Value:	118 (0x76)
Data Bytes:	12
Data Format:	Vector(float x3)
Description:	Sets the static compass reference vector

Function:	Set accelerometer reference vector
Command Value:	119 (0x77)
Data Bytes:	12
Data Format:	Vector(float x3)
Description:	Sets the static accelerometer reference vector

Function:	Reset Kalman filter
Command Value:	120 (0x78)
Data Bytes:	0
Data Format:	None
Description:	Resets Kalman filter's covariance and state matrices

User's Manual

Function:	Set accelerometer range
Command Value:	121 (0x79)
Data Bytes:	1
Data Format:	Accel range(byte, 0 for $\pm 2g$, 0x10 for $\pm 4g$, 0x30 for $\pm 8g$)
Description:	Set which range of accelerometer data to use. Higher ranges can detect and report larger accelerations, but are not as accurate for smaller ones.

Function:	Set multi reference weight power
Command Value:	122 (0x7a)
Data Bytes:	4
Data Format:	Weight power(float)
Description:	Set weighting power for multi reference vector weights. Multi reference vector weights are all raised to the weight power before they are summed and used in the calculation for the final reference vector. Setting this value nearer to 0 will cause the reference vectors to overlap more, and setting it nearer to infinity will cause the reference vectors to influence a smaller set of orientations.

Function:	Enable/disable filter
Command Value:	123 (0x7b)
Data Bytes:	1
Data Format:	Mode (byte, 0 for disabled, 1 for full kalman, 2 for gyroless-filtered orientation, 3 for fast-filtered)
Description:	Used to disable the orientation filter or set the orientation filter mode. Changing this parameter can be useful for tuning filter-performance versus orientation-update rates. When using the sensor as an IMU, it will improve performance to disable the orientation filter. When using as a AHRS, where orientation is needed, full-Kalman, gyroless-filtered, or fast-filtered should be used.

Function:	Set running-average mode
Command Value:	124 (0x7c)
Data Bytes:	1
Data Format:	Mode (byte, 0 for normal, 1 for confidence)
Description:	Selects the mode that the running-average method uses. Normal uses a static running-average. Confidence uses a running average that changes dynamically based upon the confidence factor.

Function:	Set gyroscope range
Command Value:	125 (0x7d)
Data Bytes:	1
Data Format:	Gyro range(byte: 0 for $\pm 250dps$, 1 for $\pm 500dps$, 2 for $\pm 2000dps$)
Description:	Sets the measurement range used by the gyroscope sensor.

Function:	Set compass range
Command Value:	126 (0x7e)
Data Bytes:	1
Data Format:	Compass range (byte: 0 for $\pm 0.88Ga$, 1 for $\pm 1.3Ga$, 2 for $\pm 1.9Ga$, 3 for $\pm 2.5Ga$, 4 for $\pm 4.0Ga$, 5 for $\pm 4.7Ga$, 6 for $\pm 5.6Ga$, 7 for $\pm 8.1Ga$)
Description:	Sets the measurement range used by the compass sensor.

Function:	Read tare orientation(Quaternion)
Command Value:	128 (0x80)
Return Data Bytes:	16
Return Data Format:	Quaternion(float x4)
Description:	Reads the tare orientation as a quaternion.

User's Manual

Function:	Read tare orientation(Rotation Matrix)
Command Value:	129 (0x81)
Return Data Bytes:	36
Return Data Format:	Rotation Matrix(float x9)
Description:	Reads the tare orientation as a rotation matrix.

Function:	Read rho data(Accelerometer)
Command Value:	130 (0x82)
Return Data Bytes:	9
Return Data Format:	Rho mode(byte, 1 for confidence, 0 for static), min/static rho(float), (max rho(float))
Description:	Reads the rho mode and rho data for the accelerometer.

Function:	Read rho data(Compass)
Command Value:	131 (0x83)
Return Data Bytes:	9
Return Data Format:	Rho mode(byte, 1 for confidence, 0 for static), min/static rho(float), (max rho(float))
Description:	Reads the rho mode and rho data for the compass.

Function:	Read current update rate
Command Value:	132 (0x84)
Return Data Bytes:	4
Return Data Format:	Update rate in microseconds(int)
Description:	Reads the amount of time the last frame took.

Function:	Read compass reference vector
Command Value:	133 (0x85)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Reads the single mode compass reference vector.

Function:	Read accelerometer reference vector
Command Value:	134 (0x86)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Reads the single mode accelerometer reference vector.

Function:	Read reference vector mode
Command Value:	135 (0x87)
Return Data Bytes:	1
Return Data Format:	Mode(byte, 0 for single static, 1 for single auto, 2 for single auto continuous, 3 for multi)
Description:	Reads the current reference vector mode. See command 105 for details.

Function:	Read compass multi reference vector
Command Value:	136 (0x88)
Data Bytes:	1
Data Format:	Index
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Reads the multi mode compass reference vector at index <Index>.

Function:	Read compass multi reference check vector
Command Value:	137 (0x89)
Data Bytes:	1
Data Format:	Index
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Reads the multi mode compass reference check vector at index <Index>.

Function:	Read accelerometer multi reference vector
Command Value:	138 (0x8a)
Data Bytes:	1
Data Format:	Index
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Reads the multi mode accelerometer reference vector at index <Index>.

Function:	Read accelerometer multi reference check vector
Command Value:	139 (0x8b)
Data Bytes:	1
Data Format:	Index
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Reads the multi mode accelerometer reference check vector at index <Index>.

Function:	Read gyro enabled state
Command Value:	140 (0x8c)
Return Data Bytes:	1
Return Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Reads the enabled state of the gyros.

Function:	Read accelerometer enabled state
Command Value:	141 (0x8d)
Return Data Bytes:	1
Return Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Reads the enabled state of the accelerometer.

Function:	Read compass enabled state
Command Value:	142 (0x8e)
Return Data Bytes:	1
Return Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Reads the enabled state of the compass.

Function:	Read axis directions
Command Value:	143 (0x8f)
Return Data Bytes:	1
Return Data Format:	Axis direction byte.
Description:	Reads the axis direction byte. For its meaning, see command 116.

Function:	Read oversample rate
Command Value:	144 (0x90)
Return Data Bytes:	1
Return Data Format:	Rate(1 for off, 2+ for number of samples per frame)
Description:	Reads the current oversample rate.

Function:	Read running average percent
Command Value:	145 (0x91)
Return Data Bytes:	4
Return Data Format:	Percent(float)
Description:	Reads the current running average percent. For its meaning, see command 117.

Function:	Read desired update rate
Command Value:	146 (0x92)
Return Data Bytes:	4
Return Data Format:	Update rate in microseconds(int)
Description:	Reads the current desired update rate. For more information on this, see command 103.

Function:	Read Kalman filter's covariance matrix
Command Value:	147 (0x93)
Return Data Bytes:	4
Return Data Format:	Covariance Matrix(float x9)
Description:	Read Kalman filter's covariance matrix.

Function:	Read accelerometer range
Command Value:	148 (0x94)
Return Data Bytes:	1
Return Data Format:	Accel range(byte, 0 for 2G, 0x10 for 4G, 0x30 for 8G)
Description:	Read accelerometer sensitivity range.

Function:	Read multi reference weight power
Command Value:	149 (0x95)
Return Data Bytes:	4
Return Data Format:	Weight power(float)
Description:	Read weighting power for multi reference vector weights.

User's Manual

Function:	Read multi reference resolution
Command Value:	150 (0x96)
Return Data Bytes:	2
Return Data Format:	Resolution(byte x2, number of cell divisions, number of nearby vectors)
Description:	Reads number of cell divisions and number of nearby vectors per cell for the multi reference lookup table. See command 111 for more information.

Function:	Read number of multi reference cells
Command Value:	151 (0x97)
Return Data Bytes:	4
Return Data Format:	Number of cells(int)
Description:	Reads total number of multi reference cells.

Function:	Read filter enabled state
Command Value:	152 (0x98)
Return Data Bytes:	1
Return Data Format:	Mode (byte, 0 for disabled, 1 for full kalman, 2 for gyroless-filtered orientation, 3 for fast-filtered)
Description:	Reads the value of the filter mode enabled.

Function:	Read running-average mode
Command Value:	153 (0x99)
Return Data Bytes:	1
Return Data Format:	Mode (byte, 0 for normal, 1 for confidence)
Description:	Reads the selected mode for the running-average.

Function:	Read gyroscope range
Command Value:	154 (0x9a)
Return Data Bytes:	1
Return Data Format:	Gyro range(byte: 0 for ± 250 dps, 1 for ± 500 dps, 2 for ± 2000 dps)
Description:	Reads the current measurement range setting used by the gyroscope sensor.

Function:	Set compass range
Command Value:	155 (0x9b)
Return Data Bytes:	1
Return Data Format:	Compass range (byte: 0 for ± 0.88 Ga, 1 for ± 1.3 Ga, 2 for ± 1.9 Ga, 3 for ± 2.5 Ga, 4 for ± 4.0 Ga, 5 for ± 4.7 Ga, 6 for ± 5.6 Ga, 7 for ± 8.1 Ga)
Description:	Reads the current measurement range setting used by the compass sensor.

Function:	Set compass calibration parameters
Command Value:	160 (0xA0)
Data Bytes:	48
Data Format:	Bias(float x3), Matrix(float x9)
Description:	Sets the compass calibration parameters to the given values. These consist of a bias which is applied to the raw data as a translation, and a matrix by which the value is multiplied by.

User's Manual

Function:	Set accelerometer calibration parameters
Command Value:	161 (0xA1)
Data Bytes:	48
Data Format:	Bias(float x3), Matrix(float x9)
Description:	Sets the accelerometer calibration parameters to the given values. These consist of a bias which is applied to the raw data as a translation, and a matrix by which the value is multiplied by.

Function:	Read compass calibration parameters
Command Value:	162 (0xA2)
Return Data Bytes:	48
Return Data Format:	Bias(float x3), Matrix(float x9)
Description:	Reads the compass calibration parameters.

Function:	Read accelerometer calibration parameters
Command Value:	163 (0xA3)
Return Data Bytes:	48
Return Data Format:	Bias(float x3), Matrix(float x9)
Description:	Reads the accelerometer calibration parameters.

Function:	Read gyro calibration parameters
Command Value:	164 (0xA4)
Return Data Bytes:	24
Return Data Format:	Bias(float x3), High range bias(float x3)
Description:	Reads the gyroscope calibration parameters.

Function:	Begin gyro auto-calibration
Command Value:	165 (0xA5)
Description:	Puts the sensor in gyro calibration mode. It will collect a few frames of data from the gyro to determine its bias. It will return to normal operation after this or if the sensor is reset.

Function:	Set accelerometer calibration parameters
Command Value:	166 (0xA6)
Data Bytes:	24
Data Format:	Bias(float x3), High range bias(float x3)
Description:	Sets the gyroscope calibration parameters to the given values. These consist of a bias for each gyro mode which will be applied to the data from the appropriate mode as a translation.

Function:	Read PanID (Dongle / Sensor)
Command Value:	192 (0xC0)
Return Data Bytes:	2
Return Data Format:	PanID (short)
Description:	Return this device's PanID.

Function:	Set PanID (Dongle / Sensor)
Command Value:	193 (0xC1)
Data Bytes:	2
Data Format:	PanID (short)
Description:	Set this device's PanID. Only devices with the same PanID will be able to communicate with it. For wireless sensors, this command can only be issued to a device plugged in via USB.

User's Manual

Function:	Read channel (Dongle / Sensor)
Command Value:	194 (0xC2)
Return Data Bytes:	2
Return Data Format:	Channel (short)
Description:	Return this device's channel. Note that channel values range from 11-26 (decimal), inclusive.

Function:	Set channel (Dongle / Sensor)
Command Value:	195 (0xC3)
Data Bytes:	2
Data Format:	Channel (short)
Description:	Set this device's channel. Note that channel values range from 11-26 (decimal), inclusive. An attempt to set the channel to a value outside this range will have no effect. Furthermore, only devices on the same channel will be able to communicate.

Function:	Set LED mode (Dongle / Sensor)
Command Value:	196 (0xC4)
Data Bytes:	1
Data Format:	Mode (byte)
Description:	There are two distinct LED modes for both the dongle and sensor: 0 places the sensor into normal LED mode, while 1 places the sensor into static LED mode. During normal LED mode, the dongle will flash green upon successful reception, red upon receiving no response to a transmission, and a custom defined color under all other circumstances. This is the same color that is set by the set led color command (0xEE). The sensor, under normal LED mode, flashes green upon normal reception. When the sensor is plugged in via USB, the sensor will alternate between the custom defined color and the color red while it is charging, and green once it is fully charged. While unplugged, the wireless sensor will flash red at increasingly quicker intervals once the battery level drops below a certain threshold. At all other times, the sensor will display the custom defined color. During static LED mode, both the sensor and dongle will display a constant color regardless.

Function:	Commit wireless settings
Command Value:	197 (0xC5)
Description:	Commits wireless settings to non-volatile memory. This will cause all wireless settings to persist even if the sensor is reset or power-cycled.

Function:	Read wireless address (Dongle / Sensor)
Command Value:	198 (0xC6)
Return Data Bytes:	2
Return Data Format:	Address (short)
Description:	Return this device's wireless address. Each device's wireless address is used to uniquely identify the device during communication. The wireless address is set at the factory and is cannot be changed by the user.

Function:	Read LED mode (Dongle / Sensor)
Command Value:	200 (0xC8)
Return Data Bytes:	1
Return Data Format:	Mode (byte)
Description:	Returns the currently set LED mode for the device. See the details of command 196 for detailed explanation.

Function:	Read battery voltage (Sensor only)
Command Value:	201 (0xC9)
Data Bytes:	4
Data Format:	Voltage (float)
Description:	Reads the current battery voltage as read by the battery voltage monitoring circuit.

Function:	Read battery remaining percentage (Sensor only)
Command Value:	202 (0xCA)
Data Bytes:	2
Data Format:	Percentage (short)
Description:	Reads the calculated current battery remaining.

Function:	Read battery status (Sensor only)
Command Value:	203 (0xCB)
Data Bytes:	1
Data Format:	Status (byte)
Description:	Reads status byte indicating the state of the battery. 1 represents fully charged, 2 represents charging.

Function:	Read wireless association table entry ID (Dongle only)
Command Value:	208 (0xD0)
Data Bytes:	1
Data Format:	Index(byte)
Return Data Bytes:	4
Return Data Format:	Hardware ID (int)
Description:	Return the hardware ID in the dongle's association table that is associated with the slot ID given by index.

Function:	Set wireless association table entry ID (Dongle only)
Command Value:	209 (0xD1)
Data Bytes:	5
Data Format:	Index (byte), Hardware ID (int)
Description:	Set the hardware ID in the dongle's association table that is associated with the slot ID given by index.

Function:	Read wireless channel noise (Dongle only)
Command Value:	210(0xD2)
Return Data Bytes:	16
Return Data Format:	Noise levels (byte x 16)
Description:	Returns a byte for each channel representing the level of noise on each respective channel. These values range from 0..255 with 0 meaning the channel is completely clear, and 255 meaning that it is unusably noisy. The first value returned corresponds to channel 11, and the last value returned corresponds to channel 26.

Function:	Set wireless retries (Dongle only)
Command Value:	211(0xD3)
Data Bytes:	1
Data Format:	Retries (byte)
Description:	Set the number of times the dongle will retry a failed transmission to a sensor. The maximum value, as well as the default value, is 3.

Function:	Read wireless retries (Dongle only)
Command Value:	212 (0xD4)
Return Data Bytes:	1
Return Data Format:	Retries (byte)
Description:	Return the number of retries that the dongle will attempt.

User's Manual

Function:	Read wireless slots open (Dongle only)
Command Value:	213 (0xD5)
Return Data Bytes:	1
Return Data Format:	Slots (byte)
Description:	All commands sent to the dongle to be transmitted wirelessly are buffered until previous pending transmissions complete. There are fifteen such slots, and while the dongle handles the management of them internally, no additional transmissions can be queued up if the maximum number of slots are occupied already. If sending a large batch of commands to a single sensor, it is useful to check this to make sure the transmission can be initiated.

Function:	Read reception strength (Dongle only)
Command Value:	214 (0xD6)
Return Data Bytes:	1
Return Data Format:	Value (byte)
Description:	Returns a value ranging from 0.. 255 indicating the relative strength of the most recent packet received by the dongle. The value scales linearly with distance from the dongle. Low values do not necessarily mean that transmissions

Function:	Set HID update rate (Dongle only)
Command Value:	215(0xD7)
Data Bytes:	1
Data Format:	Rate in ms (byte)
Description:	Set the update rate used by the HID (mouse and joystick) devices when used wirelessly.

Function:	Read HID update rate (Dongle only)
Command Value:	216 (0xD8)
Return Data Bytes:	1
Return Data Format:	Rate in ms (byte)
Description:	Return the currently set HID update rate.

Function:	Read version extended
Command Value:	223 (0xDF)
Return Data Bytes:	16
Return Data Format:	Version(string)
Description:	Reads the extended version string.

Function:	Restore factory settings
Command Value:	224 (0xE0)
Description:	Restores all settings to factory settings. The settings are not committed to non-volatile memory by this command, so the commit settings command will have to be used if this is desired.

Function:	Commit settings
Command Value:	225 (0xE1)
Description:	Commits settings to non-volatile memory. This will cause them to persist even if the sensor is reset.

Function:	Software Reset
Command Value:	226 (0xE2)
Description:	Resets the sensor.

User's Manual

Function:	Enable watchdog timer
Command Value:	227 (0xE3)
Data Bytes:	4
Data Format:	Timeout rate in microseconds(int)
Description:	Enables the watchdog timer with the given timeout rate. If a frame takes more than this amount of time, the sensor will automatically reset. This is useful for dealing with sensor hangs or crashes, as the sensor would reset and continue normal operation.

Function:	Disable watchdog timer
Command Value:	228 (0xE4)
Description:	Disables the watchdog timer.

Function:	Enter firmware update mode
Command Value:	229 (0xE5)
Description:	Puts the sensor into firmware update mode. This will cease normal operation until the firmware update mode is instructed to return the sensor to normal operation.

Function:	Get version
Command Value:	230 (0xE6)
Return Data Bytes:	12
Return Data Format:	Version(string)
Description:	Reads the version identifier of the sensor firmware. This will be "TSSUSB", followed by 6 digits representing the date the firmware version was created.

Function:	Set USB mode
Command Value:	233 (0xE9)
Data Bytes:	1
Data Format:	Mode(byte, 2 for Unique, 1 for FTDI, 0 for CDC)
Description:	Sets the communication mode for USB. All modes present a COM port with which to communicate with the USB device. FTDI and CDC each present a regular numbered port, whereas Unique presents a port named YEI_<serial number>. The sensor will change modes immediately.

Function:	Get USB mode
Command Value:	234 (0xEA)
Return Data Bytes:	1
Return Data Format:	Mode(byte, 2 for Unique, 1 for FTDI, 0 for CDC)
Description:	Reads the communication mode for USB, as described in the previous command.

Function:	Set clock speed
Command Value:	235 (0xEB)
Data Bytes:	4
Data Format:	Clock speed(int, Hz)
Description:	Sets the clock speed to the given value. Available settings are: 15Mhz, 30Mhz, 60Mhz. This setting does not need to be committed, but does not take effect until the sensor is reset.

Function:	Get clock speed
Command Value:	236 (0xEC)
Return Data Bytes:	4
Return Data Format:	Clock speed(int, Hz)
Description:	Reads the clock speed of the sensor's MCU. Possible settings are: 15Mhz, 30Mhz, 60Mhz.

Function:	Get serial number
Command Value:	237 (0xED)
Return Data Bytes:	4
Return Data Format:	Serial number(int)
Description:	Reads the sensor's serial number.

Function:	Set LED color
Command Value:	238 (0xEE)
Data Bytes:	12
Data Format:	Red(float), Green(float), Blue(float)
Description:	Sets the color of the LED on the sensor to the given RGB color.

Function:	Get LED color
Command Value:	239 (0xEF)
Return Data Bytes:	12
Return Data Format:	Red(float), Green(float), Blue(float)
Description:	Reads the current color of the sensor's LED, in RGB format.

Function:	Enable/Disable joystick
Command Value:	240 (0xf0)
Data Bytes:	1
Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Turns the data feed to the joystick on and off. If disabled, the sensor will still enumerate as a joystick, but the joystick will not function. This allows normal communication to occur at a faster rate.

Function:	Enable/Disable mouse
Command Value:	241 (0xf1)
Data Bytes:	1
Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Turns the data feed to the mouse on and off. If disabled, the sensor will still enumerate as a mouse, but the mouse will not function. This allows normal communication to occur at a faster rate.

Function:	Read joystick enabled state
Command Value:	242 (0xf2)
Return Data Bytes:	1
Return Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Read whether or not the joystick is enabled.

Function:	Read mouse enabled state
Command Value:	243 (0xf3)
Return Data Bytes:	1
Return Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Read whether or not the mouse is enabled.

User's Manual

Function:	Set control mode
Command Value:	244 (0xf4)
Data Bytes:	3
Data Format:	Control class(byte), control index(byte), handler index(byte)
Description:	<p>Sets the operation mode for one of the controls. The control classes are: Joystick Axis: 0 Joystick Button: 1 Mouse Axis: 2 Mouse Button: 3 There are 2 axes and 8 buttons on the joystick and mouse. The index(0 based) selects which of these you would like to modify. The handler index selects which handler you want to take care of this control. The indices are: Turn off this control: 255 Axes: Global Axis: 0 Screen Point: 1 Buttons: Hardware Button: 0 Orientation Button: 1 Shake Button: 2 For more information on these, see the section on Control Customization.</p>

Function:	Set control data
Command Value:	245 (0xf5)
Data Bytes:	7
Data Format:	Control class(byte), control index(byte), data point index(byte), data point(float)
Description:	<p>Sets parameters for the specified control's operation mode. The control classes and indices are the same as described in command 244. Each mode can have up to 10 data points associated with it. How many should be set and what they should be set to is entirely based on which mode is being used, so you should reference the section for that mode in the Control Customization section.</p>

Function:	Read control mode
Command Value:	246 (0xf6)
Data Bytes:	2
Data Format:	Control class(byte), control index(byte)
Return Data Bytes:	1
Return Data Format:	Handler index(byte)
Description:	<p>Read the index of this control's mode. The control classes and indices are the same as described in command 244.</p>

Function:	Read control data
Command Value:	247 (0xf7)
Data Bytes:	3
Data Format:	Control class(byte), control index(byte), data point index(byte)
Return Data Bytes:	4
Return Data Format:	Data point(float)
Description:	<p>Read the value of a certain parameter of the specified control's operation mode. The control classes and indices are the same as described in command 244.</p>

Function:	Set button gyro disable length
Command Value:	248 (0xf8)
Data Bytes:	1
Data Format:	Length in frames(byte)
Description:	<p>Determines how long, in frames, the gyros should be disabled after one of the physical buttons on the sensor is pressed. A setting of 0 means they won't be disabled at all. This setting helps to alleviate gyro disturbances caused by the buttons causing small shockwaves in the sensor.</p>

Function:	Read button gyro disable length
Command Value:	249 (0xf9)
Return Data Bytes:	1
Return Data Format:	Length in frames(byte)
Description:	Reads the current button gyro disable length.

Function:	Read button state
Command Value:	250 (0xfa)
Return Data Bytes:	1
Return Data Format:	Button state(byte, 1 bit per button)
Description:	Reads the current state of the sensor's physical buttons.

Function:	Set mouse absolute/relative
Command Value:	251 (0xfb)
Data Bytes:	1
Data Format:	Mode(0 for absolute, 1 for relative)
Description:	Puts the mouse in absolute or relative mode. Please note that this change does not take effect immediately, and the sensor must be reset before the mouse will enter this mode.

Function:	Read mouse absolute/relative
Command Value:	252 (0xfc)
Return Data Bytes:	1
Return Data Format:	Mode(0 for absolute, 1 for relative)
Description:	Reads the current mouse absolute/relative state. Note that if the sensor has not been reset since it has been put in this state, the mouse will not reflect this change yet, even though this command will.

Function:	Set joystick and mouse present/removed
Command Value:	253 (0xfd)
Data Bytes:	2
Data Format:	Mode(byte for each, 0 for removed, 1 for present)
Description:	Sets whether the joystick and mouse are present or removed. If removed, they will not show up as devices on the target system at all. For these changes to take effect, the sensor driver may need to be reinstalled.

Function:	Read joystick and mouse present/removed
Command Value:	254 (0xfe)
Return Data Bytes:	2
Return Data Format:	Mode(byte for each, 0 for removed, 1 for present)
Description:	Reads the joystick and mouse present/removed state. See command 253 for more detail.

Appendix

USB Connector

The 3-Space Sensor has a 5-pin USB Type-B jack and can be connected via a standard 5-pin mini USB cable.

Hex / Decimal Conversion Chart

		Second Hexadecimal digit															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
First Hexadecimal Digit	0	000	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015
	1	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030	031
	2	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047
	3	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062	063
	4	064	065	066	067	068	069	070	071	072	073	074	075	076	077	078	079
	5	080	081	082	083	084	085	086	087	088	089	090	091	092	093	094	095
	6	096	097	098	099	100	101	102	103	104	105	106	107	108	109	110	111
	7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
	8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
	9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
	A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
	B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
	C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
	D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
	E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
	F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Notes:

Serial Number: _____

Notes:



YEI Technology
630 Second Street
Portsmouth, Ohio 45662

Toll-Free: 888-395-9029
Phone: 740-355-9029

www.YeiTechnology.com
www.3SpaceSensor.com

Patents Pending
©2007-2011 Yost Engineering, Inc.
Printed in USA